Tampere University

Sara Kumpulainen

CHALLENGES AND SUCCESS FACTORS WHEN USING LEAN SERVICE CREATION IN AGILE SOFTWARE PROJECTS

Faculty of Information Technology and Communication Sciences
Master's Thesis
October 2020

ABSTRACT

Sara Kumpulainen: Challenges and Success Factors when using Lean Service Creation in Agile Software Projects.
Master's Thesis
University of Tampere
Computer Science
October 2020

Agile and Lean Software Development methods have become a very popular project management methods alongside the traditional waterfall model within the last ten years. The use of Lean and Agile methods makes it possible respond to changes market faster which is crucial for a successful software project in a rapidly changing world. Lean Service Creation (LSC) is a service design and development process that has developed around Lean Startup, Agile methods and Design thinking. The aim of the study was to find out how to ensure a smooth transition from Service Vision Sprint (SVS) to implementation in LSC projects. This thesis constructs from literature review and empirical research. The literature review reviews Agile methods in general and highlights the success factors of Agile Software Projects found through previous research. Literature review also introduces The LSC process. The results of the literature review formed the themes and the framework of the interviews. The interviews were attended by six employees of the case company who have been involved with SVS projects.

The results of this thesis are very much in line with previous studies, but some differences were also found. Differences were found regarding the importance of the documentation. Key result of the research was the importance of shared understanding among team members. Other key findings included the clarity of the documentation of the SVS outcome, the importance of technical background work in the SVS, transferring the sense of ownership to the team, collaboration of designers and developer in the SVS, inspiring leadership style and actively involving the end user.

This thesis was able to gather a list of key issues that could make the transition from the SVS to the implementation phase smoother.

Keywords: Agile Software Development, Lean Service Creation, Lean Software Development

TIIVISTELMÄ

Sara Kumpulainen: Haasteet ja onnistumistekijät ketterissä ohjelmistoprojekteissa käyttäen Lean Service Creation -menetelmää
Pro gradu -tutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Lokakuu 2020

Ketteristä ja Lean-ohjelmistokehitysmenetelmistä on tullut erittäin suosittuja projektinhallintamenetelmiä perinteisen vesiputousmallin rinnalla viimeisen kymmenen vuoden aikana. Lean- ja Agile -menetelmien käyttö mahdollistaa markkinoiden muutoksiin vastaamisen nopeammin, mikä on ratkaisevan tärkeää onnistuneelle ohjelmistoprojektille nopeasti muuttuvassa maailmassa. Lean Service Creation (LSC) on palvelumuotoilu- ja kehitysprosessi, joka on kehittynyt Lean Startup-metodologian, ketterien menetelmien ja Desing Thinking -ajatusmallin ympärille. Tutkimuksen tavoitteena oli selvittää, kuinka varmistaa sujuva siirtyminen Service Vision Sprintistä (SVS) ohjelmistokehitysvaiheeseen LSC -projekteissa. Tämä tutkimus koostuu kirjallisuuskatsauksesta ja empiirisestä tutkimuksesta. Kirjallisuuskatsauksessa käydään läpi ketteriä menetelmiä yleisesti, sekä tuodaan ilmi aiempien tutkimusten kautta löydettyjä ketterien ohjelmistoprojektien menestystekijöitä. Kirjallisuuskatsaus esittelee myös LSC-metodin. Kirjallisuuskatsauksen perusteella muodostettiin haastattelujen teemat ja haastattelurunko. Haastatteluihin osallistui kuusi case -yrityksen työntekijä, jotka olivat olleet mukana LSC -projekteissa.

Tämän tutkimuksen tulokset ovat hyvin yhdenmukaisia aiempien tutkimusten kanssa, mutta myös eroja havaittiin. Erot tämän tutkimuksen ja aiempien tutkimusten välillä havaittiin dokumentaation tärkeydessä. Tämän tutkimuksen keskeinen tulos oli tiimin keskinäisen yhteisymmärryksen tärkeys. Muita keskeisiä tuloksia olivat SVS – lopputuleman selkeä dokumentointi, teknisen taustatyön merkitys SVS -vaiheessa, omistajuuden tunteen siirtäminen tiimille, suunnittelijoiden ja kehittäjien osallistuminen SVS -vaiheessa, innostava johtamistyyli sekä aktiivinen loppukäyttäjän osallistaminen.

Tämän tutkimuksen avulla pystyttiin kokoamaan lista keskeisistä asioista, joiden avulla siirtyminen SVS vaiheesta ohjelmistokehitysvaiheeseen on mahdollista saada sujuvammaksi.

Avainsanat: Ketterä ohjelmistokehitys, Lean Service Creation, Lean ohjelmistokehitys

Contents

Abbreviations

AI,	Artificial Intelligence
JIT,	Just in Time
Jira,	Project Management Software for Teams
LSC,	Lean Service Creation
Miro,	Online Visual Platform for teamwork
MVL,	Minimum Lovable Product
MVP,	Minimum Viable Product
SVS,	Service Vision Sprint,
Trello,	Visual Tool for organizing work
UI,	User Interface
UX-designer,	User Experience designer
WIP,	Work in Progress

1 Introduction

The level of Lean and Agile methods used in software development has risen rapidly for the past 10 years in Finnish software companies. In a study done in 2012, Agile methods showed a strong position of 34% in methods used in software development in Finnish software companies. Lean methods were used by 24% of the companies usually together with Agile methods. Companies are interested in combining Agile and Lean methods, and when adopting Lean, Agile is not abandoned. (Rodriguez et al., 2012)

The most important goal for the software industry is to develop high quality and user-friendly products. Focusing on the quality of the product helps software users to adapt the product more efficiently and easily. For that, it is important to define a software development process that ensures a high-quality software product. Agile and Lean methods bring great benefits like improved team communication, enhanced ability to adapt to changes and increase in productivity. (Jain et al., 2018) There are still some challenges and limitations that can be identified such as obtaining management support and developing a large and complex software (Rodriguez et al., 2012).

Lean Service Creation (LSC) is a service design and development process that has developed around Lean Startup, Agile methods and design thinking. LSC in a customizable service design process that uses a comprehensive set of canvases to guide through the process. LSC method ensures that time and money are spend on the services and features that customers really need. There are two ways to use the LSC method: 1) Creating new services and products or 2) Cultural transformation when new ways of working and developing company culture are introduced to the company. New ways of working in cultural transformation are introduced with Agile, Lean and customer centric approaches. The aim of this thesis is to find out how the LSC process could be improved in order to make the transition from Service Vision Sprint to implementation phase smoother. The research question for this thesis is:

How to ensure a smooth transition from Service Vision Sprint (SVS) to implementation in Lean Service Creation projects?

Research on the success of a software project using Agile methods has been done to some extent, but no previous research on transition from SVS to implementation in Lean Service Creation projects has been conducted before. This thesis focuses on studying how designers, developers and facilitators perceive the success factors and challenges when

transitioning from SVS to implementation phase using the LSC method. The topic for this research was chosen by the case company that has also created the LSC method.

Chapter two is the theoretical background for this research. This chapter focuses on presenting the key findings of previous studies focusing on success factors in Agile software projects. Chapter two also introduces the Lean Service Creation method. Lean Service Creation method, together with the literature review, is the framework for the thematic interview. Based on the findings from earlier studies, the scientific background was created for this thesis.

Chapter three presents the research methods for this thesis. The chapter discusses the research question, research problem and also introduces the case company for this thesis. Interview plan and interviewees are also introduced in this chapter as well as the framework and themes for the interviews.

Chapter four reviews the results gathered from the interviews. Results of the interviews are presented through themes that were found by searching unifying and differentiating elements from the interview materials.

Chapter five discusses the research results. Key topics and findings that unite and differ with the interview results are also discussed here. Chapter five also presents the key conclusions, thesis process and future development ideas.

Chapter six provides a summary of the study, outlining the main conclusions and key findings of the study.

2 Literature review

This chapter discusses Agile development methods and how Agile development methods are utilized in today's software development. The first section reviews what agility is in practice and what an Agile approach means. The second section examines what success factors can be found if a software project is implemented using Agile methods. The third section discusses examples of Agile development methods and Lean Software Development.

The second section examines what success factors can be found from Agile software projects. Since the software projects of the research case company have also been implemented using Agile methods, the literature examines the success factors found from Agile projects. From the success factors found from the literature review, two themes were formed for the interviews.

The examples of Agile methods discussed in the third subsection, Scrum and Kanban, are so-called light frameworks for software development. Both frameworks follow the principles Agile methods and Lean. Kanban is a specific implementation of Lean. Scrum in is a specific implementation of Agile. Lean and Agile approaches share the same goals, but the way these methods are executed differs a bit. Because both of these methods are quite light weight, it is possible to mix what works the best. (CBT Nuggets, 2017)

2.1 Agile Software Development methods

The way of working in development teams using Agile methods is usually flexible and it gives room for changes. Agile methods are the opposite of the traditional waterfall model, which does not allow as much creativity or room for possible changes. (Jain et al., 2018) The waterfall model follows a strict phase concept where each step is performed in order, limited to the original requirements and the design is carried out at the beginning of the project. If a customer approves the product to be deployed, and there have not been any changes to the requirements during the development process, the project could be completed within the time and budget. In general, however, projects implemented with a waterfall model take longer to produce a product that meets the requirements. They respond to rapidly changing requirements poorly, because only project manager monitors the process of the project as it progresses. (Jain et al., 2018) In contrast, Agile software projects are more people-oriented in nature than process-oriented. This means that individual people and interactions are valued more than processes and tools. (Agile Manifesto, 2001) Agile methods are customer-oriented and customer satisfaction is being highly valued by being in touch with the customer in every stage of the sprint. (Kassab et al., 2018)

Agile methods have become a very popular project management methods alongside the traditional waterfall model in the field of software development. Agile methods enable the development of the high-quality software on a fast schedule.

Core values of Agile Manifesto are the following (Agile Manifesto, 2001):

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

The development model is Agile if development is incremental (small releases with a fast schedule), responsive (the customer and the developers are cooperating continuously), straightforward (method itself is easily assimilated and easy to modify and well documented) and adaptive (possible to make last minute changes) (Abrahamsson et al., 2002).

There are also following principles behind the Agile manifesto (Agile Alliance, 2001):

1. The highest priority is to keep a customer satisfied through early and continuous delivery.
2. Welcoming all the changing requirements, even in the late development for the customer's competitive advantage.
3. Delivering working software frequently. Preferring a shorter timescale from couple of weeks to couple of months
4. Developers and business people must work together daily throughout the whole project.
5. Building the project around motivated people. Trusting the people to get the job done and giving them an environment and needed support.
6. The most efficient and effective method for spreading information to and between the development team is face to face communication.
7. Dg The primary measure for progress is a working software.
8. Agile processes favor sustainable development. All the sponsors, developers and users should be able to keep up the work pace indefinitely.
9. Constant attention to good design and technical excellence improves agility.
10. It is essential to maximize the amount of work not done.
11. Self-organizing teams are a base for best architecture, requirements and design
12. On a regular basis, the team reflects how to improve effectiveness and tunes its behavior accordingly

2.2 Agile approach

As already stated above, Agile approach is an iterative and incremental method that favors small releases. Each step is performed in short iterative development sprints with a maximum time frame of one month. (Schwaber & Sutherland, 2017; Jain et al., 2018) Agile Software Development process consists of five steps: Requirement analysis, design, development, testing and maintenance. Agile Software Development process is described in Figure 1.

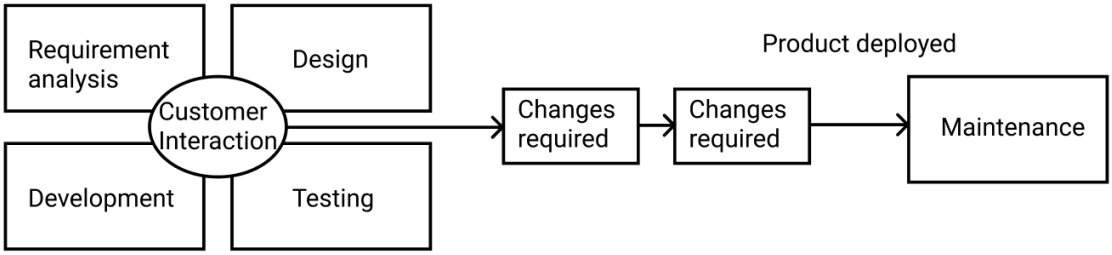


Figure 1. Customer centered Agile methodology. Picture adapted from (Jain et al., 2018).

Requirement analysis is a phase where all the customer's needs and requirements are gathered to functional and non-functional requirements. In the design phase, the architectural side of the future software is created. This will determine how the final software will appear to the customer and how it will work. Development phase is for implementing the working version of the software. Testing is the final step in the software development process before the working version of the product is handed to the customer. Software testing is a process to find bugs and errors. This step is done for ensuring and validating that the software has all the needed functionalities. If any errors or bugs are found after the deployment, this usually leads to changes in the software product, meaning in the phase Maintenance. This usually improves the performance of the software and it becomes more adaptable to the changing environment. (Kassab et al., 2018)

2.3 Success factors in Agile software projects

In this section, two studies are reviewed that focus on the success factors in Agile software projects. First study is conducted by Chow & Cao (2008) where the authors systematically analyzed prior research on success factors in Agile software projects. The other research was conducted by Kelle et al. (2015) and it focuses on the social success factors in Agile Software Development. Agile Software Development methods are originally applied to or considered to be successful on smaller projects or teams. Scaling up Agile software methods are considered to be challenging. Both of the aforementioned studies tried to unveil success factors of Agile software projects. (Kelle et al., 2015)

Chow & Cao (2008) determine, drawing from earlier research, that failures and problems in Agile software projects fall into four categories: organizational problems, human problems, process problems, and technical problems. Success factors, on the other hand, fall into five categories: organizational success factors, people-related success factors, process-related success factors, technical success factors, and project-related success factors. Kelle et al. (2015), based on literature and qualitative interviews constructed a conceptual model of social factors that may have effect on the success of Agile projects or software projects. Project size was also included as a success factor candidate in this research. Kelle et al. (2015) tested the model on a set of 40 projects and they compared a total of 140 project members, Scrum Masters and product owners.

Building on previous research, Chow & Cao (2008) define that the characteristics of success that describe the perception of project success can be considered; quality, scope, time and cost. Quality means either delivering a good product or the project is seen as successful. Scope means that all the requirements and objects are met. Time as a success characteristic means that delivery of the project happens on time. Cost as a characteristic of success means that delivery happens within estimated effort and cost.

A total of 55 potential success factors that could affect an Agile software project were found in the previous literature by Chow & Cao (2008). The factors were further categorized into a list of 39 distinctive features.

Because the research of Chow & Cao (2008) was exploratory in nature, they performed a reliability analysis for each factor to ensure the highest possible level of reliability. This was done to determine the extent to which each factor was related to the other factor. The results identified 12 factors that became the main hypotheses for the study. The hypotheses were united on the basis of them being a critical success factor for Agile software projects in some of the four areas of success: quality, scope, time, and cost. Main hypothesis described on the Figure 2.

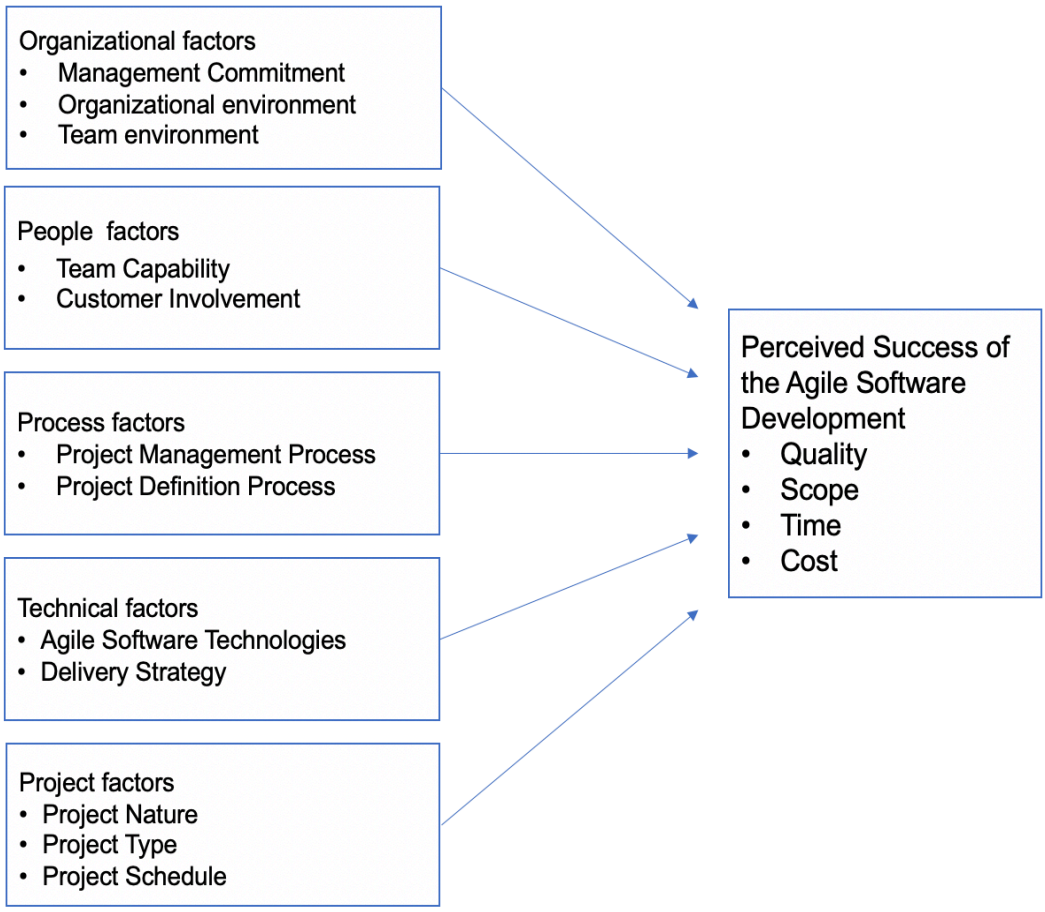


Figure 2. Research model for success factors in Agile software projects (Chow & Cao, 2008)

Chow & Cao (2008) found the following critical success factors: The correct delivery strategy, proper practice of Agile software engineering techniques and high-caliber teams. Chow & Cao (2008) also found the following to be somewhat critical success factors: Good Agile management process, Agile-friendly team environment and a strong customer involvement. The delivery strategy is correct if the team is delivering the most important features first and the delivery of the software is regular. Agile software engineering techniques are proper if the coding standards are well-defined upfront. Integration testing, right amount of documentation and rigorous refactoring activities also are part of Agile software engineering techniques. Project management process can be seen as a success factor if its requirement process, project process, and configuration management process is Agile-oriented. It is also important to have a strong communication focus and good process tracking mechanism while managing a projects' process. Team environment is Agile if teams remain small, coherent and self-organizing. Projects should not have multiple independent teams and teams should have the collocation of the whole team. Customer involvement as a success factor is giving the whole authority to the customer and the customer having commitment and presence during the project. Things that affect the

capability of the team are appropriate technical training to the team and highly motivated team members with high competence and expertise.

The study conducted by Kelle et al. (2015) chose the candidate success factor based on prior research. These candidates include leadership style, communication style, value congruence, degree of adoption of Agile practices and project size. Leadership style was addressed in two terms, transformational and transactional leadership style. Transformational leaderships stand for a leadership style that is adaptive and focuses on long term commitment. These types of leaders motivate, inspire, express visions and engage their followers with emotional involvement. Transactional leadership style focuses social transactions where expectations and rewards are stated clearly and there exists a short-term focus. Because the main reasons for project failure are often the lack of effective communication and misunderstandings, informational communication helps to build trust, creates more shared values and creates strong interpersonal relationships. These can be considered to be crucial success factors in Agile Software Development. Agile projects are known to have turbulent and changing environments and therefore this thesis stated that informational communication and communication style are more important than communication frequency in both small and larger projects. Similarity in values and goals enhances interpersonal relationships and is essential to be effective and efficient. If values, goals and targets differ between the members of the team, it can cause value diversity. That can increase conflicts in relationships, decrease satisfaction and negatively affect software's performance.

Kelle et al. (2015) developed and validated a new conceptual model to examine relationships between various candidate success factors and Agile project success. The new model was developed incorporating qualitative interviews, qualitative validation and prior research. In the first exploratory phase, the model has five candidate success factors: Transformational leadership, Communication style, Value congruence, Degree of Agility and Project size.

In the second phase Kelle et al. (2015) validated the conceptual model and determined the relative importance of each success factor. Before testing the conceptual model, linear regression analyses were conducted to examine relationships between the project success and proposed success factors. Study found correlations between all the other success factors and project success except project size. The most significant relationship existed between transformational leadership, value congruence and degree of agility which the study suggests to the most critical success factor for Agile project success.

Kelle et al. (2015) found that the size of the project did not determine if the Agile project was successful. The most important predictors for the success of an Agile project were found to be the degree of Agile practices, value congruence and transformational leadership. The researchers then divided the participating projects into two groups, the one that scored mediocre on the success factors and one that scored high on the success factors. This was done to assess to what extent the identified success factors influence the project success. Then the average scores of these groups were examined in relation to project success. Their study showed that degree of agility and value congruence had larger differences compared to transformational leadership. Kelle et al. (2015) inspected the average scores on projects success for the groups of projects where none, only 1 out of three, 2 out of three or all 3 factors scored high. This showed the researchers that to maximize the project success, all of the three factors should be given attendance. Their study confirmed that social factors have an impact in success or failure in projects.

Based on the research results of Chow & Cao (2008), the success factors of an Agile software project can be said to be proper use of Agile software engineering techniques, the correct delivery strategy and high-caliber teams. Chow & Cao (2008) also found the following to be somewhat critical success factors for Agile projects; good Agile management process, Agile-friendly team environment and strong customer involvement. Based on the previous research of Kelle et al. (2015) the size of the project did not determine if the Agile project was successful. They found the following to be critical success factors for Agile project: degree of Agile practices, value congruence and transformational leadership.

2.4 Examples of Agile development

2.4.1 Scrum

The main idea of Scrum is to divide bigger wholes and products into smaller parts that could possibly be delivered with a shorter schedule. Scrum is a framework that has been used for over two decades to help software teams to improve their teamwork. (Agile Alliance, 2018) Scrum encourages teams to learn through experiences, self-organize and improve through winnings and losses (Atlassian, 2020). Ensuring Transparency gives benefits to software development teams. Scrum gives the ability to focus on value, to experiment and to react to new knowledge quickly (Wykowska & Wykowski, 2018). Scrum has values, principles and practices that allows teams with different sets of backgrounds to deliver products and services with a short schedule. Scrum Team investigates every functionality item after its delivery and decides afterwards what features will be implemented based on learning, feedback and the possibility to minimize risks and reduce

waste. This cycle is repeated as long as the team gets to deliver the final product that meets the customer's needs. Scrum helps to (Scrum Alliance, 2020):

- Get quick feedback
- Improve continually
- Adapt the changes quickly
- Deliver the product with shorter schedule

The positive aspects of Scrum are the maximum return on investment guaranteed for the business, because Scrum Teams constantly create the primary parts of functionality in cycles. Scrum helps businesses to (Scrum Alliance, 2020):

- Innovate faster
- Improve constantly
- Adapt to changes quickly
- Deliver products faster

The Scrum Team

A Scrum Team usually consists of Product owner, Development team, and Scrum Master. Scrum Teams are self-organizing with multi-skilled team members. When teams are self-organizing, team members decide themselves the best practices to work to finish the job rather than someone outside the team directing the work. Multi-skilled teams have all the conditions to succeed in their goals without any factors outside the team, because the Scrum model is designed to optimize flexibility, creativity and productivity. (Schwaber & Sutherland, 2017)

Because a Scrum Team delivers products incrementally and iteratively, it also maximizes the possibility for feedback. Incrementally obtained "Done" version of the product enables the working version of the product to be always available. (Schwaber & Sutherland, 2017)

It is the Product owner's responsibility to maximize the value created by the Development Team. Product Owner is also responsible for managing the product backlog, which consist of the following actions (Schwaber & Sutherland, 2017):

- Clear description of the work items on the backlog.
- Ordering the backlog items the way that goals and missions will be reached.
- Optimizing the development team's value of work.
- Ensuring that backlog items are clear and visible to all.
- Showing the next work tasks to team members.
- Ensuring that development team understands the work items on the backlog.

If someone wants to make any changes to the backlog, it has come through the product owner. In the end, the Product Owner is responsible for product backlog items apart who is executing those. (Schwaber & Sutherland, 2017)

The Development Team is responsible for delivering the "Done" at the end of each sprint, which is a potentially releasable increment of the product. The Development Team consists of professionals, usually programmers and designers who manage and organize their own work. Because this self-organizing is empowered by the organization, it optimizes the Development Team's overall effectiveness and efficiency. The optimal team size should remain small enough to remain Agile, but large enough to complete all the task from the backlog within each sprint. The Development Team has the following characteristics (Schwaber & Sutherland, 2017):

- Self-organizing skills. The team decides the ways to deliver potentially releasable functionality.
- Cross-functionality. Developers have all the necessary skills to create an increment.
- No titles. Team members have no titles, regardless of the work tasks being done.
- No sub-teams regardless of the domains that need to be addressed.
- Accountability responsibility to team as a whole despite the specialized skills team members may have.

The main responsibility for Scrum Master is to maximize the value that Scrum Team delivers by helping all Scrum Team members to understand Scrum theory, practices, rules and values. Scrum Master can be called as a servant leader who serves and guides the Scrum Team. Scrum Master serves the Product Owner, Development Team and organization. (Schwaber & Sutherland, 2017)

The Scrum Process

The most important part of the Scrum is the Sprint, which is the time frame when the releasable functionalities from backlog are created. Sprint usually lasts from two weeks to one month and they last throughout the whole development effort. When the sprint comes to a conclusion, a new sprint starts immediately after (Schwaber & Sutherland, 2017). Each sprint has a goal and could be described as a small project itself. So that sprint goal would not be endangered, any big changes are not done during the sprint. (Schwaber & Sutherland, 2017)

A common sprint consists of the Sprint planning, Daily Scrums, development work, Sprint Reviews and The Sprint Retrospective (Schwaber & Sutherland, 2017). Scrum process is described on Figure 3.

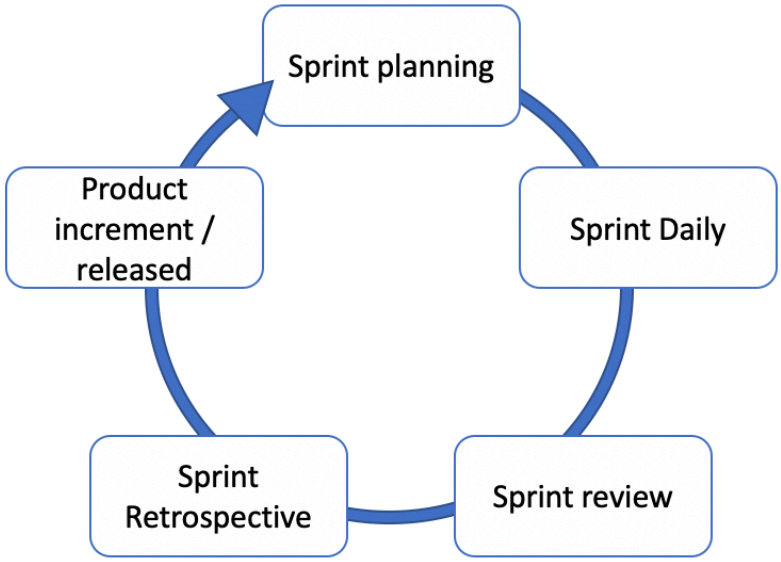


Figure 3. Scrum process (Kurnia et al., 2018)

Sprint planning as a phase the whole team takes part of. Planning the sprint consists of two main discussion topics (Kurnia et al., 2018):

1. The review of the previous activities on product backlog and defining the sprint goal.
2. Team defining the activities of following sprint and choosing the product backlog items to be developed.

Sprint daily or daily Scrum meeting is a meeting involving the team members who are involved during the sprint. Idealistically this meeting lasts approximately 15 minutes and is meant for team members to state their current work and summarize the targets to be achieved. (Kurnia et al., 2018)

Sprint review is a phase that involves also the stakeholders outside the development team. Sprint review is meant to demonstrate the results the team has achieved during the sprint. The next sprint planning is based on customers' or stakeholder's feedback.

Sprint retrospective is a phase happening between Sprint review and next Sprint planning. The goal of retrospective is to improve communication between the team members for a smoother collaboration for the following sprint. (Kurnia et al., 2018)

2.4.2 Lean

Lean thinking and Lean Software Development have become more popular when developing software and digital services for its goal to improve customer satisfaction and quality. Lean thinking is a management philosophy which focuses on removing unnecessary and unproductive activities. Its goal is to create synergies within the organization and on the entire value chain. (Janes, 2015)

The principles of Lean thinking and Lean Software Development are based on the success of Lean manufacturing from a Japanese car company Toyota at the end of 1940. The traditional mass production was questioned because it produced thousands of the same kinds of cars for Japanese market, but the market was not big enough for selling all these cars. Since people did not have much money after the war and cars had to be cheap, Toyota had to find a way to produce as inexpensive cars as mass-produced cars but in small quantities. The Toyota Production System was not fully recognized until the oil crisis in the 1970's. Because of Japanese companies adapting the Toyota Production System's features, they were able to hit the world market with dramatically low pricing. Japanese companies were using a new approach called Just-In-Time that was then adopted by European and American companies as well to remain competitive. (Poppendieck & Poppendieck, 2003) Just-In-Time manufacturing and Production refers to producing units needed, in the time needed, and in the quantities needed. (Lulu, 1986)

The core idea of Lean is to maximize customer value and minimize waste at the same time. That means that more value is created to the customer with fewer resources (Lean Enterprise Institute, 2020). Customer value is a high priority for Lean organizations, and they focus its key processes to increase it continuously. To accomplish zero waste, the focus of management is changed from optimizing separate technologies, assets and vertical departments to optimizing the flow of products and services. This is done by optimizing the flow through entire value streams that flow horizontally across assets, technologies and departments to customers. This creates processes that do not require so much human effort, space, capital or time to make products and services with smaller costs. Information management becomes more accurate and simpler and it comes easier to respond to rapidly changing customer desires faster and with high variety, quality and low cost. (Principles of Lean, 2020)

Ultimately, only the customer can define the value. That means specifying the value from the standpoint of the end customer by product family. Mapping the value stream means identifying all the steps in a value stream for each product family. Every step that does not create value, should be eliminated. So that the product flows smoothly towards the

customer, the value-creating steps should occur in tight sequence. Establishing a pull means letting the customer pull the product from you, which is pulling the value from the next upstream activity. The previous steps are repeated until the state of perfection is reached. That means creating perfect value with no waste. (Lean Enterprise Institute, 2020; Womack & Jones, 2003) The five-step process for taking Lean method into use is described in Figure 4.

Figure 4. Lean principles (Lean Enterprise Institute, 2020).

The Principles of Lean Software Development

Due to the domain variability, for example comparing manufacturing and software development, it is challenging to adopt Lean Principles to software development. Because the value is not limited to a certain time-limited effort, the concept of value is not straightforward (Razzak, 2016). Lean production is aiming to reduce the timeline from receiving an order from a customer to collecting the cash by removing all the waste that gives no value. That is the core idea of Lean production. Lean Software Development has the same goal with eliminating the waste, but the timeline from order to delivery differs a little. The timeline starts from order and stops when the software is deployed that meets the needs of the customer. (Poppendieck & Poppendieck, 2006) The matter of waste also differs between manufacturing and software development, because the work items in software development are more intangible. In the manufacturing world, inventory can be seen as waste, but in software business the partially done work is seen as waste. The main goal in Lean Software Development is to implement the principles from Lean manufacturing into a software development model. That is, reducing waste in a system and creating a higher value for the customer. (Razzak, 2016)

Poppendieck and Poppendieck (2006) present the seven principles of Lean Software Development in their book *Implementing Lean Software Development: From Concept to Cash.* The seven principles are Eliminate waste, Build Quality In, Create Knowledge, Defer Commitment, Deliver Fast, Respect People and Optimize the Whole.

Eliminate waste. This includes understanding what value is. In software business the value may change, because the customer's idea about what they want will often change, when they see the software in action. One form of waste in a software development is for example partially done work that quickly becomes outdated and it may get lost. It could also hide quality problems and tie money. Other form of waste if called the "churn" or "requirements churn". This means the amount of work and time that it is used for testing and fixing that usually takes a lot longer than the actual development. This is very much associated with large amounts of partially done work and when requirements are done long before coding. The third and the biggest form of waste in software development are extra features that were not really needed in the first place. Developing extra features cost a lot and add complexity to the code base. This makes it more expensive to maintain the software. Because customers often don't know what they want, Lean Software Development needs a process that develops less code with more value and then develops the next important features. (Poppendieck & Poppendieck, 2006)

Build Quality In. Testing should happen as soon as possible and fix the defects as soon as they are found. Defects in a queue are seen as waste and the ultimate goal of Lean is to have no waste at all. Automation and refactoring also builds quality in. Test-driven development (TDD) is an effective approach to improve code quality. Writing unit tests and acceptance test before writing the code helps to detect defects sooner. If a test fails, the problem will be fixed, or the code will not be used. (Razzak, 2016)

Create Knowledge. Projects that undergo constant changes and respond to changes in the market are more likely to be successful than projects that strictly follow the early requirements. Projects with early releases of minimum features receive important feedback so they are continuously improving. Lean organizations learn about the product under the development and codify the knowledge for future use. (Poppendieck & Poppendieck, 2006)

Defer Commitment. Planning is important for software projects but sticking to a plan makes the software weak because plans change easily. Planning is still a good learning experience and it is needed to create high-level architecture design. When developing

early features, decisions that lock any critical design decisions should be avoided. One of the best software design strategies is to leave options open, so any critical decisions are possible to made as late as possible. (Poppendieck & Poppendieck, 2006)

Deliver Fast. It should be possible to deliver software so fast that the customer does not have time to change their minds. The old belief is that to achieve high-quality, you have to be careful and slow down your speed. Nowadays, the belief that speed and quality are incompatible is false. Companies competing against time have a big advantage over competitors. Eliminating a huge amount of waste saves in costs and lowers defect rates. Teams still need to maintain quality in their work to maintain the speed. Lean development teams have engaged people who are trusted and are able to make decisions and changes without permission. This leads to teams continually improving their processes and responding to customers wishes faster. (Poppendieck & Poppendieck, 2006)

Respect People. In software development, businesspeople should be trusted that they are able to do their job, even if you have just started in your job. Companies that respect their people usually create great leaders and that creates successful products. To sustain competitive advantage, companies also need to ensure that teams have all the technical expertise they need to accomplish their goals. (Poppendieck & Poppendieck, 2006)

Optimize the Whole. Lean should be implemented to the whole value stream. Value stream should not be broken into silos, because the overall system will probably be sub optimized. (Poppendieck & Poppendieck, 2006)

2.4.3 Kanban

Kanban is a Japanese word for visual signal. The Kanban method is a set of practices and principles that are applied to an existing process. (Kanban University, 2020) Kanban - board is a tool that is designed to help Agile projects to visualize the work, reduce the amount of work in progress and maximize the efficiency. Figure 5 shows an example of Kanban board. Kanban boards use cards, columns and continuous development to help technology and service teams work with the right amount of work and to get their work done. Because a Kanban board helps to visualize the amount of work, it is easy for all team members to "stay on the map" of the project phases. (Atlassian 2020)

The history of Kanban dates back to the Lean industry, but the "Kanban method", originally defined by David Anderson helped to bring Kanban to software development industry as well as the service industry (Atlassian 2020). Anderson originally developed the Kanban method for information and service work in 2005 by combining elements from

the works of different researchers and management specialists, such as W. Edwards Deming, Eli Goldratt, Peter Drucker and Taiichi Ohno (Kanban University, 2020).

Over the years, the Kanban method has evolved to a set of practices and principles. In collaboration with the wider Kanban community, the Kanban method was created as it is known today. (Kanban University, 2020)

The core principles of Kanban are the following (Kanban University, 2020):

- Start with what you do now
- Agree to pursue evolutionary change
  Initially, respect current roles, responsibilities & job titles
- Encourage acts of leadership at all levels

Core practices of Kanban are the following (Kanban University, 2020):

- Visualize
- Limit WIP
- Manage flow
- Make policies explicit
- Implement feedback loops
- Improve collaboratively, evolve experimentally (using models and the scientific method)

Kanban board

Kanban is a method used in Agile Software Development and a popular tool for managing projects (Atlassian, 2020; Nakazawa & Tanaka, 2015). Kanban boards are used to visualize the task statuses and the overall flow of the work tasks. The board is split into a vertical state and it represents the state of the tasks. Tasks are represented as cards on the board and they are placed to the board to a stage representing the task status. Tasks are then moved inside the board from left to the right. A task usually includes a concise description of the user requirements and names or icons of the person responsible doing these tasks. (Nakazawa & Tanaka, 2015) Kanban boards can be broken into five different components: visual signals, columns, WIP limits, commitment points and a delivery point. Visual signals are the first thing that people see when looking at a Kanban board. They could be either sticky notes, tickets or other visual cards. Team members use these cards to write their projects, user stories or work items onto. Usually one card holds one work item. These cards help both team members and other stakeholders to see and understand what others are working on. Columns in Kanban board represent a specific activity. These activities combined are called the “workflow” along which the cards are moving.

Workflows are e.g. “todo”, “in progress” and “complete”. Work In Progress (WIP)-limits mean the maximum amount that any column should have cards at any time. WIP-limits are critical for maximizing the flow and exposing any barriers in workflow. WIP- limits can be seen as an early warning that team members could be loaded with too much work. Commitment point is where team members and customers can put ideas that can be picked up when work starts on the project. Delivery point is the team’s workflow’s endpoint, which usually means that the product or service is handed to the customer. The ultimate goal for the team is to put items from commitment point to delivery point as fast as possible. (Atlassian, 2020).

Kanban boards can be either physical or digital. Due to globalization, distributed product development industry and distributed teams, digital boards came to use. Digital boards have some advantages. It is possible to view past tasks and they offer statistics and it is possible to see how much time people are spending on a task. The advantage of a physical board is that they offer better visualization, more accessibility and they motivate teams more. (Bacea et al., 2017) Even if both implementations have their advantages and disadvantages, operating Kanban methods having the advantages of both require high price instrument or high synchronization cost. (Nakazawa et al., 2017). Illustration of a Kanban board on Figure 5.

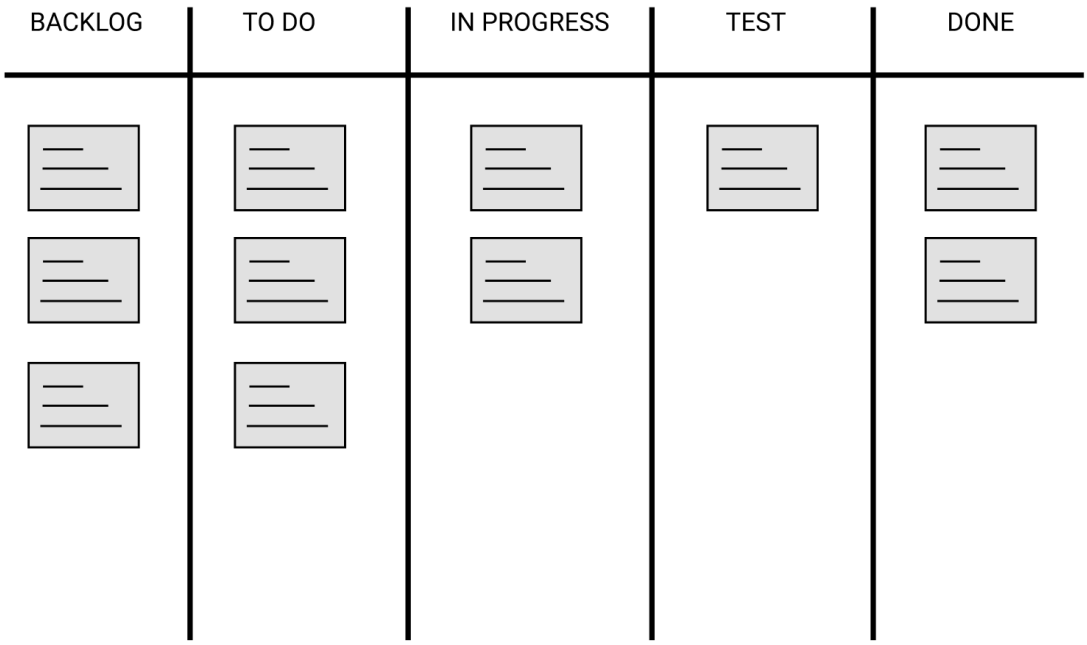


Figure 5. Illustration of a Kanban board.

2.5 Lean Startup

Lean Startup methodology was developed in 2011 by Eric Ries. This methodology has become widely popular in the innovation field. The Lean Startup method was created to help startup managers to avoid resource or financial expenses when creating a product that is not guaranteed to be successful on the market. The main goal for the Lean Startup method is to prevent maximum risk and continuously improve the product and adapt marketing strategy by entering early into the market and it is used as a tool to expand the client base. The Lean Startup process could take place in multiple iterations. This process includes development of the viable product and testing. Testing helps to notice all the comments and wishes from the consumer. Lean Startup is a method used both startups and also large companies i.e. General Electronics, Airbnb, LinkedIn, Dropbox and Toyota. (Veretennikova & Vaskiv, 2018) Figure 6 describes The Lean Startup process (The Lean Startup, 2020).

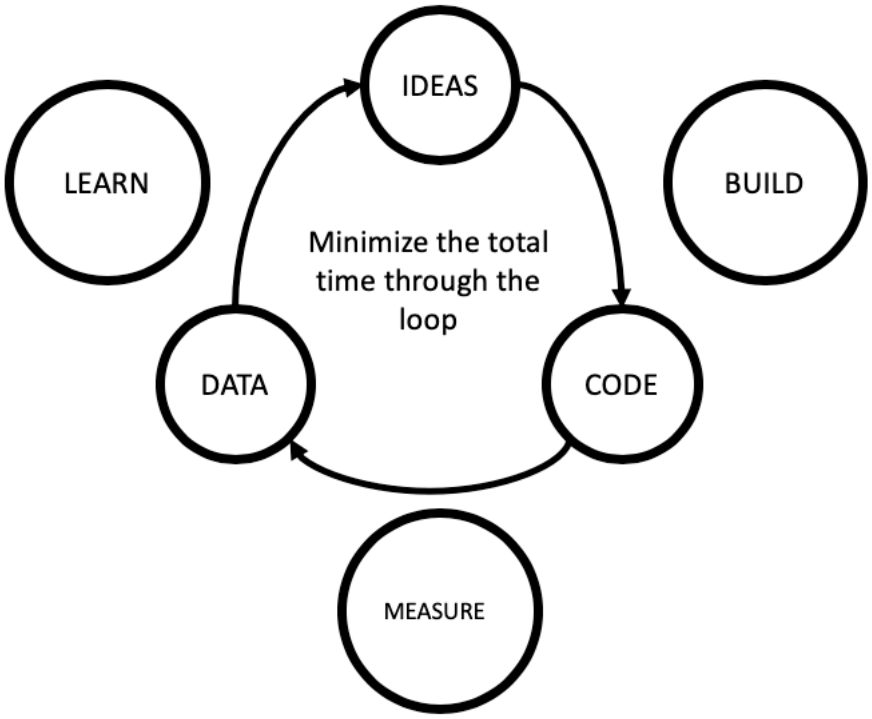


Figure 6. The Lean Startup process (The Lean Startup, 2020).

Principles of Lean Startup methodology (The Lean Startup, 2020):

1. Entrepreneurs are everywhere
2. Entrepreneurship is management
3. Validated Learning
4. Build-Measure-Learn
5. Innovation Accounting

Eric Ries, the founder of Lean Startup methodology describes startup as a "A human institution designed to create new products and services under conditions of extreme uncertainty." This means that people engaged in these efforts are entrepreneurs, even if they do not see themselves as one. *Entrepreneurship is management* means that if innovation can be managed at a startup, a new type of management is needed that can handle the unpredictability startups usually have. The process of capturing the knowledge a new startup makes is called *validated learning.* This is when the startup device and run experiments using the scientific method that either prove or reject the hypothesis that is behind the business model. This iterative process is called *Build-Measure-Learn*. Innovation accounting means how progress should be measured. The progress measurement should not be the same as in established business, so the right signals are being fed back into the feedback loop. Validating and rejecting the hypothesis should also happen correctly. (Brem & Fredriksen, 2016)

2.6 Design Thinking

Design is a process of creating artifacts that solve problems. Many fields like products, fashion, architecture, software development and engineering are familiar with design. In the software development business, design means creating a software product that does a job that users want done. Software designers have gathered a lot of practical wisdom that can be seen in different design principles. Designers purposely support practices, worlds and identities of the users. Software designers manifest principles such as separation of concerns, abstraction, modularity, layering, wholeness, utility, resiliency, beauty and timelessness. These principles are a guide to build products that have the most useful and meaningful user experience in their user community. (Denning, 2013)

Since 2010, Design Thinking offers new models of processes and toolkits that help designers and multidisciplinary teams to improve, accelerate and visualize creative processes. Design Thinking is also seen as a motor of innovation for designers. The aim for Design Thinking models is to replicate non-linear and iterative character of the process and also to replicate the alternation between diverge and convergence moments that are inherent in problem solving related to design. (Clemente et al., 2016).

Design Thinking is a concept which expresses how experienced designers approach design problems (Clemente et al., 2016). Its goal is to focus the design around user's concerns, interests and values. Design Company IDEO has a Design Thinking Philosophy that emphasizes design as a team sport with the following principal values; Many eyes, Customer viewpoint and Tangibility. Many eyes as a value can be seen as a team with

diversified experience. Different areas of expertise e.g. engineering, human factors, communications, graphics, ethnography and sociology can give unique perspective to other team members to things they might not see. Second value, customer viewpoint means involving the ultimate user experience into design. Design team could visit customers interviewing and observing what they really do so they can catch reactions from extreme "stress" cases. Denning (2013) also bring third principal value, Tangibility, which means learning from feedback and reactions after customers have tried built prototypes or mockups.

Nakata and Hwang (2020) state that the mindsets behind Design Thinking are human centeredness, abductive reasoning and learning by failing which are all essential and distinctive to design thinking. Human centeredness is considered so important to design thinking that some are using a fuller name, "human centered design thinking". Design thinking attempts to emphatically solve user issues as experienced. This includes an entire range of emotional, embodied and material events. Customer preferences are translated into experiences that are meaningful and memorable and they become part of the innovation themselves. Abductive reasoning makes the ideation process in project work visible. It asks rather what should be, than what already is. Abduction is based more on assertion than evidence and it is taking creative leaps. Abduction cuts the ties to the apparent and factible in order to imagine possibilities. The third mindset, learning by failing is about framing failure as a necessary part of learning. To arrive sooner to better and effective solutions, failing is something not to be scared of. Always looking for safe answers to errors and always making appropriate choices might not solve or satisfy underlying needs. Risk taking and mistakes, on the other hand, can lead to unexpected solutions. (Nakata & Hwang, 2020)

2.7 Lean Service Creation

This section introduces The Lean Service Creation. Lean Service Creation (LSC) is a Creative Common licensed service design and product development process developed by Futurice, the case company of this thesis. Lean Service Creation was chosen as one of the theoretical frameworks for this thesis as the aim of the study was to find out how to ensure a smooth transition from Service Vision Sprint (SVS) to implementation in Lean Service Creation projects. The first subsection views the definition of LSC and what are the goals of LSC. The second subsection introduces the LSC canvases. The third subsections introduce the principles and practices of LSC Manifesto. The fourth subsection goes through the LSC phases.

2.7.1 The Definition of Lean Service Creation

The LSC process is a compilation of smart tools that help teams to systematically and flexibly create new services. LSC has been developed around Lean Startup, Agile Development and Design thinking underpinning the expertise Futurice has from thousands of software projects. LSC is a customizable service design process that uses a comprehensive set of canvases to guide through the process. LSC method ensures that time and money are spent on the services and features what customers really need. LSC could either be used to create new services and products or for cultural transformation when new ways of working and developing company culture are introduced to the company. New ways of working in cultural transformation are introduced with, Agile, Lean and customer centric approaches. In its simplicity, LSC is a guide for creating services at all stages of a project, from early ideation to code finishing. (Futurice LSC, 2019)

The first version of LSC was crafted in 2013. Back then, the LSC process was all about creating digital services from business, user and tech viewpoints with a holistic process and toolkit, but nowadays the process includes a fourth dimension as well for societal and environmental impact of services as well. (Futurice The LSC Handbook, 2019)

LSC canvases are designed to act as an innovation checklist and system for teams to organize the answers easily. The canvases are designed to be tangible on purpose. The posters can be stuck on a wall for easier team collaboration, communication, and working in the same physical space. The LSC method is a great set of tools for teams to develop themselves. They encourage teams to act differently and push concrete results. LSC method makes teams to give and receive quality feedback, experiment and fail faster, iterate and learn and tackle problems step by step. The canvases also help to turn abstract ideas into tangible ones. LSC also encourages team members to show, listen and talk to each other. (Futurice The LSC Handbook, 2019).

The aim of the LSC method is also to encourage teams to reflect their old ways of thinking. The teams and people who have embraced LSC, have been gone through the following changes; They beware functional silos and create multidisciplinary teams. Every expert is given an equal voice. They also become more holistic and understanding the bigger picture becomes easier. People and teams learn to embrace uncertainty and become more open and curious minded while working. They always validate building, measuring and learning. The LSC method also encourages co-design with customers and have fun while working. (Futurice The LSC Handbook, 2019)

2.7.2 The LSC Canvases

Lean Service Creation offers a curated set of canvases that help to go through all the necessary steps for creating successful products services (Nevanlinna et al., 2018). The Lean Service Creation process has natural checkpoints for teams to help decision making. The LSC canvases can be embraced by multiple business fields. They can be helpful, for example, to production teams that are not so familiar to customer-centric and experimental workflows. For engineers, canvases could be used to combine design thinking into Agile work methods and architecture planning. They are a great reminder for experienced service designers for more technical issues and business questions. Businesspeople can use the canvases to combine operational customer-centric core to their work. Start-ups could get great help from the canvases to communicate different ideas and plan for the future. LSC method helps research and development organizations to create a shared language and more customer-centric and Agile way of working. (Futurice The LSC Handbook, 2019)

2.7.3 The Principles and Practices of Lean Service Creation

Authors of Open Source Tools for Change Agents, Nevanlinna et al. (2018), have set five basic principles as LSC Manifesto. These five principles act as a guide to live and work by for multidisciplinary teams creating a new service.

All for the Team, Team for All is the first principle of LSC manifesto. In order to creativity to flourish in a multidisciplinary team, it is needed that team members trust and care for each other and to be open to new people and ideas. Team members should also systematically work together to find team's own best practices and routines. Hierarchy and airtight social groups are a big barrier to creativity and innovation.

Love the Problem, not the Solution is the second principle of the LSC manifesto. This principle encourages finding the problem worth solving, which stands for identifying a substantial and scalable business opportunity. (Nevanlinna et al., 2018). LSC manifesto encourages to love both your customer's problems and your organization's business problems.

No matter what you do, be transparent about it is the third principle of LSC manifesto. Every decision, creation and rationale should be made understandable and abstract things tangible. Transparency helps others to co-create, give feedback and help you with their expertise which is the key to take your work forward.

Never stop iterating, never stop learning is the fourth principle of Lean Service Creation Manifesto. There is always a chance to learn when creating a new service. Learning often means that it is required to go back and iterate. Project plans should not be too detailed but leave room for learning and iterating. When the teams gets smarter and learns, also goals and targets should be iterated and changed.

See the Forest, See the Trees, Spot the Squirrel is the fifth principle of LSC manifesto. Understanding and seeing the bigger picture is the key to understanding how your work fits in the project and what is the required impact of your work. Knowing the bigger picture gives clarity between the details of your work and broader activities.

2.7.4 The Lean Service Creation Phases

The LSC method can be divided into three phases; Service Vision Sprint (SVS), Defining and Building the MVP and Growing it Huge. In the first phase, SVS, the validated value proposition and tested business case is created. The second phase, Defining and Building the MVP (Minimum Viable Product) is the phase where the first launchable product is defined. In this phase, the MVP is also validated and tested by users. After going live with the service, comes the final phase, Growing it Huge, which is all about continuing iterating so the service could grow huge. The canvas templates can be found in Appendix A and Appendix B. LSC process is illustrated in the Figure 7:

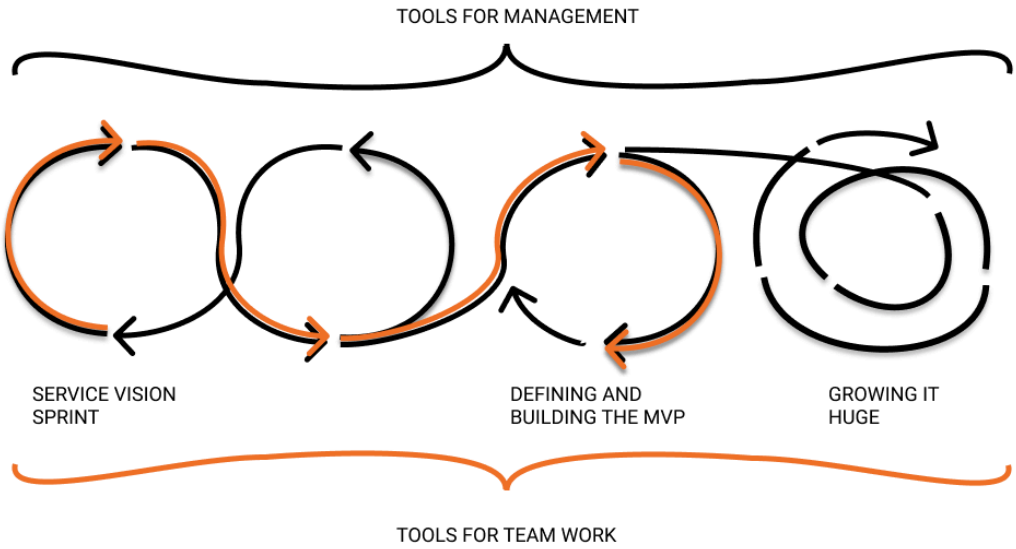


Figure 7. LSC phases (Nevanlinna et al., 2018).

SVS is the core of LSC. After completing this phase, it is usually clear what the service is all about. This is the phase where a validated value proposition is created and there is a tested business case. SVS is often called the design phase even though it is more than that. It is an iterative process that gets the multidisciplinary team on the same page. In this phase, the ownership is given to all team members. The work becomes meaningful when the team meets the end users and understands their needs. In this a service is built together with the end user. (Nevanlinna et al., 2018)

Table 1. Canvases in SVS phase (Nevanlinna et al., 2018)

The Phase	**The Purpose**	**The Canvases**
Business Objective	The common goal for the team is set to function effectively. The goal must be supported by everyone.	Business Objective and Context – **Should not be skipped** Immersion
User needs	This is where the insight interviews are made. To fully understand the needs, motives and values. There comes great value when the whole team participates in the insight interviews.	Customer Grouping Script Creator Insight
Ideation	To find an idea that fulfills the business needs, user needs and society's needs. It also has to be technically feasible.	Ideation Sandbox Idea Accelerator
Concepting	To form a full concept from a good idea.	Rational Concept Sheet Concept Sheet Impact Optimiser Customer Engagement – **Should not be skipped**
Business Model	To see if the idea/concept makes any sense from the perspective of cost and revenue.	Business Model & Market Size
Validation	Testing the value proposition. This is done, not only to prove to have a great concept but also to prove the concept might not work.	Feasibility Study Evaluating the Concept Validation – **Should not be skipped**

Defining and Building the MVP

Defining and building the Minimum Viable Product (MVP), or other way to call it "Minimum Lovable Product" (MLP) is the next phase after completing the SVS phase. In this phase, the building of the service starts and the first launchable version of the service is defined. This means building the leanest or most minimal version of the product or service that customers will love. Minimum means that if a feature can be left for later, it should

be left for later. MVP should still always deliver enough value for a company's own strategic goals and business goals. This phase should be completed using all the team tools that LSC offers, i.e. weekly canvases and retrospectives. (Nevanlinna et al., 2018).

Table 2. Canvases in Defining and Building the MVP phase (Nevanlinna et al., 2018)

The Phase	The Purpose	The Canvases
Defining and Building the MVP	To ensure that the essence of all the phases is captured.	Minimum Lovable Product MVP Backlog What to Measure

MVP should always bring real value to the end user, so it has to be lovable. This is also the key for validating the value proposition or product/market fit. Making a prioritized list of the aspects and values that must be delivered to the customer should be done. The idea of building the MVP is to build a version that lets the builders understand early what a good service could look like. Typical pitfalls that might occur are adding too much or too less and that user needs are not discussed as much as the service features. (Nevanlinna et al., 2018)

After this phase, the minimum value the MVP will deliver to the customer has been defined. Definition of done means that it can be confidently described, what needs to be built next, and why. (Nevanlinna et al., 2018)

Growing it Huge

After going live with the service, real hard data is gotten from the users, actions and business. Continuing iterating becomes important at this point so the service could grow bigger. In this phase, The LSC workflow applies as well. Assumptions should be validated, and experiments conducted. Changes to the service is encouraged to make and measure the effects afterwards. If the changes did not bring desired effects, going back to the previous version is encouraged. (Nevanlinna et al., 2018)

Table 3. Canvases in Growing it Huge phase (Nevanlinna et al., 2018)

The Phase	The Purpose	The Canvases
Growing it Huge	To grow the service to make it huge	Growth Hacking

3 Method of Research

This chapter goes through the research method. The first section outlines the research problem and the case company's problem. The second section introduces the research question and the aim for this thesis. The third section introduces the case company for this research. The fourth section goes through the interview plan, interviewees, the interview frame and interview themes.

3.1 Research problem

The aim of the study was to find out how to ensure a smooth transition from Service Vision Sprint (SVS) to implementation in Lean Service Creation projects. The case company had an interest in the handover between SVS and software development phase. There was interest in knowing if the handover contained any gaps and how the handover could be improved in order to make it more painless. The research question for this thesis is:

How to ensure a smooth transition from Service Vision Sprint to implementation in Lean Service Creation projects?

The other research question is:

How Agile practices are used in relation to Lean Service Creation projects?

The first research question was selected by the case company. The case company wanted to know if there are any ways to improve the transition from SVS to implementation phase in Lean Service Creation projects or make the transition smoother. The second research question was formed because Agile Software Development methods are used in the case company.

3.2 Case Company

Futurice Oy as a case company is suitable for the research question, as Lean Service Creation is a method originally developed by the case company and the method is used extensively inside the company. The case company was chosen for this research because the writer of this thesis works there as a software developer and had a personal interest in Lean Service Creation.

Futurice Oy is a Finnish company founded in 2000, whose main products are web and mobile software, consumer applications and business solutions. Futurice's goal is helping

its customers to create innovations through digital service design, Agile Software Development and Lean organizational change. Futurice employs more than 600 people and is present in Finland, Sweden, Norway, Germany and United Kingdom. (Futurice, 2020)

The core services of Futurice include Cloud and Data Platforms, Transformative Experiences, Innovative and Data-Enabled Organizations, Software Development, and Intelligent Services and Ecosystems. Futurice has clients across multiple different business sectors. These clients include for example Terveystalo, Fortum, BMW, Nokian Tyres, and Kesko. (Futurice, 2020).

3.3 Interview Plan and Conducting the Interviews

The aim was to get the widest possible range of designers, developers and project managers who have been involved with LSC to participate in the research interviews. The goal was to get at least ten interviewees from across the company to take part of the study to get views from as many different angles as possible. The final number of interviewees was six; three designers, two developers and one facilitator. A message was posted to the general channels of the communication application, which explained the aim of the research and clarified what kind of interviewees are needed for the research. The end of the message, it was asked to contact the researcher privately if there was any interest to take part in the research. This message helped to find three interviewees and the three remaining were obtained by asking directly about their interest to participate in the study, since the author knew about their participation in the Lean Service Creation process.

The time to gather the interviewees was short, which was two weeks. Interviews began in at the end of August 2020 and were conducted over the next two weeks. Four of the interviews were held as video conference and remaining two face-to-face. Face-to-face interviews were held at safety distance as they were conducted during Corona time. The interview material was then analyzed, and the results were written over the next two weeks.

3.3.1 Interviewees

Interviewees of this research are the employees of the case company. The Interviewees consist of designers, developers and one facilitator who has a tech background as a developer. The above-mentioned Interviewees were chosen because the different views from designers, developers and facilitators could give valuable feedback about the Lean Service Creation. Interviewed designers consisted of both UX designers and service designers. Four of the interviews were conducted in Finnish and then translated to English. Other two remaining were held in English.

3.3.2 Interview Frame and Themes

Based on the literature review, interview themes and questions were created to answer the research problem as well as possible. The interview framework also included questions for gathering the necessary background information. The following themes and subthemes were found from the literature review related to success factors in Agile software projects and Lean Service creation:

Start: Thank you for taking part in this interview. The purpose of this thesis is to find out how the transition phase between Service Vision Sprint and Defining the MVP phase could be improved for it being more efficient. In addition, it will be studied how Agile practices are used in relation to Lean Service creation projects. This interview lasts around 60 minutes, and it is being recorded. Is this okay? If you have any questions during this interview, feel free to ask. Also, this interview can be stopped anytime if you feel like it. Do you have any questions? Let's start the interview.

Background Information: 5min

- What is your current role?
- How long have you been working in the software business?
- What is your experience on Agile methods?
- How have you been involved with Lean Service Creation?

The first theme in this interview frame, Transition phase from design phase to development phase, is formed on the basis of a research question. The research question is; How to ensure a smooth transition from Service Vision Sprint (SVS) to implementation in Lean Service Creation projects. The purpose of this theme is to find out how transition between Service Vision Sprint and implementation phase works in general and could it be improved to be more efficient. The interview frame originally used the term transition phase, but after the first two interviews, the term handover was used more in the interviews, as the interviewees found it more familiar.

1. Transition from LSC to implementation phase
    - Transition phase in practice
    - The good sides in transition
    - The bad sides in transition
    - Improving the transition

- The communication

Interview question based on the first theme, Transition phase from design phase to development phase, are the following:

1. What happens in the SVS phase?
2. What happens in defining and building the MVP phase?
3. When the actual development phase starts in relation to design?
4. What happens in the transition phase?
5. How does communication work between the designers and developers?
6. How does communication work between project managers and the rest of the team?
7. What good sides you can you find in transition from SVS to development?
8. What negative sides you can you find in transition from SVS to development?
9. How transition could be improved? (how handover could be improved)
10. What other parts of LSC phases would you like to take part on?

The second theme, Service Vision Sprint, is based on The Lean Service Creation section in the literature review. This theme focuses on the customer role, end user role, validating and LSC canvases.

2. Service Vision Sprint
    - The Customer Role
    - End user role
    - Validating

Interview question based on the second theme, Service Vision Sprint, are the following:

11. What kind of canvases are you familiar with working in LSC projects?
12. What kind of role does the end user/customer have in Service Vision Sprint?
13. How Validating works in LSC?
14. What is the most important LSC canvas that should not be skipped regarding the transition phase? Why?
15. What is the least important LSC canvas that should not be skipped regarding the transition phase? Why?

The third theme, Defining the Minimum Viable Product, is also based on The Lean Service Creation section in the literature review. This theme focuses on the LSC canvases. This theme's questions were very much combined with the previous theme, Service Vision Sprint, since it also dealt with LSC canvases. LSC canvases were gathered to support the interviews. However, there was not much discussion about canvases, but mainly what canvases were familiar to the interviewees.

3. Defining the Minimum Viable Product
    - The Canvases

Interview question based on the third theme, Defining the Minimum Viable Product, are the following:

16. What kind of canvases are you familiar with working in LSC projects?
17. What kind of Teamwork canvases are you familiar working with LSC projects?

The fourth theme, Communication, is based on the chapter in the literature review, Success factors in Agile software projects. This theme focuses on how communication between the designers, the developers and project manager work in general and what leadership style interviewees are familiar in agile projects.

4. Communication
    - Communication between designers and developers
    - Communication between project manager and rest of the team
    - Leadership style / management process

Interview question based on the fourth theme, Communication, are the following:

18. How does communication work between project managers and the rest of the team during the development phase?
19. What kind of leadership style/management process are you familiar with working in LSC related projects?

The fifth theme, Agile Practices, is based on the section in the literature review, Success factors in Agile software projects. This theme focuses on the agile practices the interviewees are familiar with agile projects. Interview questions focus on team environments, tools and practices.

5. Agile Practices
    - Environment
    - Tools
    - Agile software engineering techniques

Interview question based on the fifth theme, Agile Practices, are the following:

20. What kinds of team environments have you been working in LSC related software projects?
21. What Agile software engineering techniques are you most familiar with?
22. What kind of a development team do you usually have in LSC related software projects? How many designers and how many developers?
23. What kinds of tools or practices have you used from LSC related software projects? Which practices or tools would you highlight and why?

4 Results

This chapter introduces the results of this thesis. The interviews revealed several ways how the handover phase could be improved in the Lean Service Creation process. By reviewing the interview material, it was possible to find similarities, which were used to thematize the interview material. The first section introduces the overview of the themes and their sub-themes created from the interview material. The second section reviews the interview results of the first theme *Service Vision Sprint.* The third section presents the results of the interviews under the second theme *Handover*. The fourth section reviews the interview results of the third theme *Shared Understanding*. The fifth section introduces the interview results of the fourth theme *Agile Practices*. The sixth section reviews the interview results of the fifth theme *Stakeholders*.

4.1 Overview

The themes remained almost the same as the themes of the interview body, but sub-themes emerged from the interviews and were added under the main themes. The 5 main themes and sub-themes created from the theme design are the following.

1. Service Vision Sprint
   1.1 Service Vision Sprint Process
   1.2 Technical Background Work
   1.3 Prototype
   1.4 The Team Composition and Roles

The first main theme, *Service Vision Sprint*, is derived directly from the interview frame. The sub-themes under Service Vision Sprint consisted of issues that arose in several interviews as going through this main theme. Service Vision Sprint was chosen as one of the main themes for its being the core of The Lean Service Creation process.

2. Handover
   2.1 Timeframe between Service Vision Sprint and implementation phase
   2.2 Documentation of the Service Vision Sprint outcome
   2.3 Minimum Viable Product
   2.4 Strong Hypothesis and User Stories
   2.5 Handover kickoffs & Internal Planning sessions
   2.6 Ownership of the Product

Handover, the second main theme, was also formed directly from the interview frame. Handover means the actions and issues involved when the project is handed over the production team after completing the Service Vision Sprint. Handover became one of the main themes as it is strongly connected to the research question, and it is an integral part

of the Service Vision Sprint. Sub-themes were selected based on the common handover-related topics that arose in the interviews.

3. Shared Understanding
   3.1 Understanding the bigger picture
   3.2 Sharing understanding

The third main theme, *Shared understanding*, was raised as one of the main themes as its importance emerged several times during the interviews. *Understanding the bigger picture* and *sharing understanding* became the sub-themes for this theme. Sub-themes were formed past the questions as they emerged during the interviews.

4. Agile Practices
   4.1 Team environment
   4.1 Leadership style
   4.2 Communication

The fourth main theme, *Agile Practices,* was also formed directly from the interview frame. The sub-themes for this main theme are *Team Environment, Leadership style* and *Communication*. These sub-themes were also formed directly from the interview frame. The sub-themes address topics that were seen to have an impact on the success of an Agile success project in a literature review.

5. Stakeholders
   5.1 Customer role
   5.2 End user role

The fifth main theme, *Stakeholders,* was picked as one of the main themes, as the role of stakeholders emerged on several occasions in the interviews. *Customer role* and *End user role* are the sub-themes for this theme. *Stakeholders role* was originally a sub-theme for Service Vision Sprint main theme in the interview frame, but previously mentioned sub-themes raised it to its own main theme as they stood out repeatedly in the interviews**.** Sub-themes are formed past the questions as they emerged during the interviews.

4.2 The Service Vision Sprint

Service Vision Sprint (SVS) is the first main theme. The SVS is the core of the Lean Service Creation process (Futurice Lean Service Creation Handbook, 2019). The outcome

of the SVS is a validated value proposition and business case that has been tested thoroughly. (Nevanlinna et al., 2018)

4.2.1 The Service Vision Sprint Process

Interviewees described SVS as an iterative process that creates a vision of the concept and supports the development of the Minimum Viable Product. SVS consists of several workshops and bigger kickoff type workshops where the whole business domain is gone through with the customer. Typically, SVS is used for service design. Another use case is organizational change programs if an organization wants to change the way it operates.

Interviewees described SVS also as a tool for building a common understanding. The point is also to get a shared understanding between the client and the team so that it is clear what is being done and what are the things that create business value in the project. In the citations below, interviewees describe their vision of the SVS process:

> *SVS in an innovation journey where you are constantly building assumptions, validating assumptions, learning and iterating from there, so it's a smooth process where you are always iterating and gaining more market evidence.* - Facilitator 1

> *It's not just about design, it's about increasing overall understanding so that the whole team that joins the project understands the entire business and industry.* -Designer 1

4.2.2 Technical Background Work

The importance of technical background work was often highlighted in interviews, so it became a separate sub theme within SVS main theme.

Because SVS is usually run by designers, it is sometimes referred to as the design phase. One interviewed designer stressed that the SVS is both service design and technical design. Sometimes SVS is implemented on two levels. Service design is carried out at the design angle and then a technical study or technical specification is done which gives understanding to the next phase.

Technical background work that is done in the SVS is case-specific. Technical background work is usually done by software architects and it involves more higher-level things, such as architectural matters. The aim is to resolve these matters as early as possible, so that sensible solutions can be produced. According to ons interviewee, technical

background work might involve for example technical architecture workshops with the client's own technical persons, where client's systems and dependencies are examined. The lead developer is usually responsible for architecture and the technical vision.

One interviewed designer pointed out that SVS should serve three levels; business, design and technology. One of the designers interviewed stressed that when resolving case-specific issues, most often the biggest and most important things are found on the technical side. If these technical issues are not taken into account at an early stage, it could cause problems later in the customer path. The interviewed designers below underline the importance of technical background work in SVS phase:

> *What is also worth considering is the customer's ability to understand different technological solutions. Designers don't always have the best ideas, like "how to automate this". It's really important to get tech people involved in this.* - Designer 2

> *If we do not understand technical challenges or organizational challenges that are often related to technology or old legacy, that is, if the problem field is not understood there as a whole, solutions may be harder to make reasonable at later stage.* - Designer 1

4.2.3 Prototype

In interviews, the prototype was described as being usually static pictures made by designers that are meant to represent the vision of what the service will look like and what problems are being solved for the customer.

The general idea of the MVP is created in the SVS. Definition of the MVP is one of the most important outcomes of the SVS. Definition of the MVP is a coherent vision with enough evidence to support the development of the MVP. Vision also includes goals, such as how much money this service could save for the customer. This vision could be built as a prototype which can be validated by users and stakeholders. Prototype and its importance as an SVS outcome were something that both designers and developers underlined in the interviews. The interviewees below talk about the importance of the prototype in the definition of the MVP:

> *It would be smart if there would usually be some prototype stage first, where that prototype would determine what can be iterated first before we start building anything extremely tangible and big into that MVP stage.* - Developer 1

> *We get the general idea of MVP in the SVS, but in order to start development, the developers need some kinds of wireframes, prototypes, UI design to work through.* – Designer

The interviews revealed that doing a proper background work in SVS and building a good prototype motivated the developers. It also allowed the team to iterate the concept before starting to build the MVP. The interviewees below describe how good prototype increases motivation towards the project:

> *Personally, the most motivating thing is if the team has produced a good groundwork and prototype at the SVS stage, like it is so cool that I will be building this!* - Developer 1

4.2.4 The Team Composition and Roles

Based on the interviews, the team composition in the number of designers and developers varies from project to project and it depends on what skills are needed. Developers are more involved in projects where the customer has purchased the entire project and SVS is integrated into it. The development team can be a team of up to ten people or as minimalistic as two developers. Some projects also include data scientists, business designers and AI-specialists. UI-design and UX-design starts in the MVP phase or when there is the development team already involved. In the following quote, the interviewed developer recommends that the role of design should also be included after the SVS:

> *Personally, I recommend that the design role is not terminated after the SVS phase. Especially at the beginning of the MVP phase there is a need to iterate and interview users and at the beginning of development there is a need for UI-skills.* - Developer 1

Involving the production team and UX designers as much as possible in the SVS was seen as beneficial for more efficient information sharing. Designer interviewed mentioned that participating also in the SVS and not just supporting the development team was excellent for giving in-depth background information about customer needs. The interviewees tell in the following quotes how meeting the users makes the job more meaningful and clearer:

> *This makes it much easier to do a project and also more meaningful when you understand the target and have met the end users face-to-face. It's then really easy in the*

handover to explain why we are doing this and also to explain and argue through our own experiences. - Designer 3

At the beginning of the current project, there was an SVS phase where these canvases were used to do these exercises with clients. SVS was visible in this project in a way that vision became clearer and maybe all stakeholders became clear. - Developer 2

However, because involving the entire production team to SVS was seen to be extremely expensive, it was understood that it's not possible in most occasions. A smoother transition from SVS to production side was considered to help to share understanding more effectively. One internal kickoff was not seen enough for shared understanding. The interviewed designer below suggests a process where the outputs of SVS would be reviewed with the SVS team:

When we know who is in the production team, we could do the first sprint planning where the SVS team would be starting that project so that there wouldn't be such a big gap. Even if that first sprint doesn't start next week, the production team would get the hang of it and could chew through SVS and discuss how they could build the first sprint. -Designer 1

4.3 The Handover

Handover study refers to the actions and issues involved when the project is handed over the production team after completing the SVS. Handover became one of the main themes as it is strongly connected to the research question, and it is an integral part of the SVS. This section reviews the common handover-related topics that arose in the interviews.

4.3.1 Documentation of the Service Vision Sprint

The interviews revealed that documentation of the SVS was seen as a small challenge in the handover. The needs do not always meet on the technical side and on how the service designer has documented them. Found from the interviews, the outcome that comes from SVS was sometimes found to be on so called insight level that it could challenge creating shared understanding among the team. If the outcome was documented only on an abstract level, i.e. from the business perspective, it was considered being less inspiring for the developers to start their work and the understanding the big picture might not be as well understood. These challenges might occur when the development team is not participating in the SVS. For more painless handover, so-called sprint zero was suggested to take a place after the SVS, where the output of the SVS would be dismantled into smaller

parts. The interviewed designers below express how it might be harder for developers and UX designers to jump into the project if the outcome of the SVS is documented in so called insight level:

> *If the outcome of the SVS is documented in so called insight level, from a business a business perspective for example, it might be harder to jump into a project with a mindset of "okay, let's do this", versus if all of the insight, business problems and user problems are in as concrete level as possible in the MVP* - Designer 1

> *Sometimes it's tricky for UX-designers to start working strictly from service design outcome, they need some sort of wireframe so they can start working from user stories.* - Designer 2

Interviewees told that building a prototype can be difficult if the concepts are so obscure that it is difficult to get a grip on them. One of the interviewed designers wished for an LSC canvas that would bite even deeper into the concept. The interviewee told that when the client has a really clear vision about the service, there could be some canvas to gather really concrete ideas and not just inevitably logically arranged things. The interviewed designer below gives an example what concrete ideas could be gathered from the customer to the canvas:

> *We want everything to be blue, we want to have a timeline like this, we have a vision that this is really cool. When those are not collected very systematically during the SVS, it (concept) could remain very abstract.* - Designer 3

Too abstract documentation of the SVS Sprint was also seen challenging for bringing shared understanding between the customer and the project team. Some interviewees told that the concept sometimes felt unclear and that it was lacking some concreteness. Shared understanding between the client and the designers is not always conveyed if the terminology is familiar only to the designers. The interviewees told that because some customers may not be familiar with service design terminology, they might not always fully understand what has been said or why something is done a certain way. The interviewed designer below suggests that more colloquial language could sometimes be used instead of service design terminology:

> *We should try to make those terms more colloquial. Of course, it depends on the customer. Some have their own service designers involved in the project, but then there*

are clients who don't know what service design is so you might not want to use the most challenging terms but speak really clearly. - Designer 3

4.3.2 The Minimum Viable Product

The MVP stands for "Minimum Viable Product" that is the leanest or the most minimal version of the product or service that customers will love (Futurice, The LSC Handbook, 2019). One of the interviewed developers told that the point of the MVP is to give something real to customer or end users and ask them what they think about the MVP and is there something that could be done better and how they really use the service or the product. This is the phase where the real iterative development starts when there already are some real users who get some value from the MVP.

Some interviewees told that the MVP needs to be something concrete and bring some value and it can no longer be at the prototype level like a vision. At its best, the MVP is something really small or light that could be released with a short schedule. One interviewee stated that it does not need to be anything finished, but it should meet the needs of the user quickly or that it solves some specific issue for some specific users. MVP could also be done as a pilot, where some limited user group tests the MVP when the MVP is not yet released publicly. If the MVP is published publicly it is possible to see how value is converted into money. In the following quote, the interviewed developer underlines how interesting it is for the team to see the return of the value:

This is when you actually publish something, market it a little and follow it for a few months. Then you see why users pay for this, keep using it, or if the pricing is right. This is really interesting stuff in the MVP phase. - Developer 1

The MVP backlog represents the current understanding of what the MVP should consist of and will be iterated and reprioritized constantly. The interviewed facilitator below highlights why the list of hypotheses is important for a good MVP:

The MVP team knows what the purpose is of what we are building and what we are trying to experiment or what we are trying to learn so they can build and prioritize accordingly. - Facilitator 1

4.3.3 Strong hypotheses and User Stories

A strong hypothesis was also found very important for successful handover from SVS to implementation phase. Writing down strong hypothesis and writing user story maps together with the team was seen important for good a MVP outcome. In the quote below, the interviewee's example of a strong hypothesis:

> *This app with these features will allow us to have X more sales in Y time.* - Designer 2

One interviewee mentioned that it is not stated in the LSC that writing down hypotheses is necessary or how to write user stories. It was seen helpful if a team has a member with a good Agile background who knows how to get the useful information out of everyone for user stories and knows how to put hypotheses right into place. The interviewed designer and facilitator below talk about challenges in knowledge transformation:

> *If the team does not have such a person (with Agile background), sometimes the outcome of MVP comes out weak and then the scope starts growing. Then we cannot deliver in time because we haven't really followed the foundations of Agile which means you have to have a solid hypothesis you are testing with this product.* - Designer 2

> *Sometimes information gets lost, sometimes even the interpretation of the hypothesis and the assumptions could become a classic knowledge transition challenge.* - Facilitator 1

4.3.4 Timeframe between Service Vision Sprint and Implementation phase

Found from the interviews, the transition time from SVS to the development phase may vary. The transition period could be from one week to several months. Implementation does not always start quickly after the SVS ends, but there can be a longer period where the company is waiting for customers' budgeting. Challenges may arise if the transition period is extended to many several months. The information may become outdated or the SVS report cannot be found when the production finally starts. In that case, it might not be easy for the production team to go through all the relevant SVS materials. Facilitator below express a concern about too long transition period and what difficulties it might bring:

Then information is outdated, and people can't remember what the details were and there might be people who have left the company. That is a huge challenge. -Facilitator 1

4.3.5 Internal Planning Sessions and Handover kickoffs

The SVS starts with customer kickoffs, but internal kickoffs are also held if the production team has not been involved in the SVS. Handover kickoffs are approximately half-day to day lasting workshops where the results of SVS are gone through precisely with the SVS team. According to the material collected from the interviews, the internal kickoffs include a preliminary presentation of the project, people presentations, practices, and preparations for the kickoff with the client.

Some interviewees had conflicting views on how well knowledge and understanding are shared in internal kickoffs. One interviewee said that internal kickoffs offer very good specifications for development work when the SVS team is doing the handover and involved in the project after the kickoff. In the following quote, the interviewed developer explains how handover kickoffs work in practice:

We went through SVS material where practically the whole story was told from the beginning, like "this is the need, this problem is being solved". We also looked closely through the customer path with the canvases. – Developer 1

One of the interviewees said that internal kickoffs are pretty lightweight and that they sometimes are left as a superficial presentation of the project. It was seen to be due to the fact that people still have ongoing projects, the mindset has not yet settled on starting something new. The perspective of one of the interviewed designer's was that people may have a hard time gaining understanding and deep interest in the future with just one meeting. It was suggested that once the production team is formed, an internal planning would be held where the groundwork would be done before the customer kickoff so that everyone in the team is aware what is to come. In the quote below, a designer from the interviews gives an example of what an internal meeting should go through:

It would be more of a Sprint planning style event where you would brief out the outcome of the SVS and what is the level of concreteness. It would then be used to form stuff into a Jira or somewhere else. -Designer 1

One interviewee told that their project team did a preliminary sprint planning internally already in the SVS, because the client was not previously familiar with Agile projects.

This sprint planning was to consider what was needed from the client, and what needed to achieve in future sprints. The interviewee told that planning the future sprints was done beforehand because the project had only eight sprints and it was pretty clear to the team what was needed to achieve. After the SVS, the implementation started the following week.

4.3.6 Sense of ownership

According to the Lean Service Creation Handbook (Futurice, 2019), SVS will get the team to the same page and give them a sense of ownership. The interviews revealed that problems may arise if the sense of ownership is not transferred to the team. One interviewee told that the team doing the SVS is usually really excited about the product and believes the idea of the MVP but sometimes after the SVS phase, when the project goes more towards implementation, team members are no longer so much involved in creative building and shaping the idea but are rather executing a list of things from the backlog. It was seen important that the team is actively steering the direction, rebuilding assumptions, is very focused on validating and less focused on implementing. The interviewed facilitator explains below how management style could hurt the ownership:

> *It's important that the team carries the vision and ownership of the product, nothing hurts the ownership more than a manager that comes and says: "no we are doing it this way and not that way" without giving a proper explanation.* – Facilitator 1

4.4 Shared understanding

4.4.1 Understanding the Big Picture

Understanding the bigger picture behind the project was something that both the developers and the designers found to be essential for successful handover from SVS to implementation phase. Shared understanding in this context means that everyone in the project team understands what the service or product is being made, and what are the ultimate reasons behind building this service or product. This means that the whole team also understands the ultimate needs for the customer and end user. Below the interviewed designer highlights the importance of shared understanding in the handover:

> *In an ideal world, there is already an understanding of what we are going to do, so that when the production starts, we will not be wondering "what is this thing".* – Designer 1

Understanding the big picture was seen as the most motivating thing for interviewees when moving to a new project. When people understand the entire business, industry and user problems, and are on the same page, the work becomes more meaningful. Kickoffs are a great way to accomplish that, as the interviewed developer mentions:

> *Personally, the most motivating thing is that when you understand what you are doing and not just jumping into the project with a mindset of "Hey let's fix this app or build this app and we are helping this organization but I don't know why but let's do it anyway!". After the kickoff you feel like "so cool that I'm building this thing!"* – Developer 1

4.4.2 Sharing Understanding

Some interviewees brought up that there should not be a strict transition phase between SVS and implementation phase. Some interviewees stated that if there is a clear transition, it is important to share a list of hypotheses, MVP backlog and transcription of the SVS phase in the handover. In terms of knowledge sharing, the role of the people leaving the project was considered important and it was seen as beneficial that these people could still participate in the sprint reviews by giving feedback. In the quote below, the interviewed facilitator shares a vision of a good handover where the understanding is shared to the team:

> *If there is a transition, it's important that the vision, the evidence and the assumptions of the product are transitioned to the team. And you'll have something like MVP backlog prioritized.* – Facilitator 1

Interviewees told that sometimes kickoffs may focus on too technical things that could cause information overload to developers joining the project. Interviewees described that the purpose of handovers and kickoffs was to be the motivator for the team when joining the project and hand over the keys of the project to the entire team. From the developer's perspective these internal kickoffs were quite heavy because of information overload of the technical details and long days. It was suggested that starting the handover little less factually could ease the sharing of understanding. The interviewed developer points out below that there should be more attention paid towards motivating the team and that they understand the "why" behind the service or a product rather than focusing on technical stuff:

> *We don't necessarily need to jump to such a practical level like we often do in handovers. Setting up accounts and technical stuff can be done later and little by little as*

the project processes. We should start a little slower and avoid really long and heavy all-day workshops. – Developer 1

If the handover happens only by giving a document that came out of the SVS or handing a prototype for testing, it might not motivate all the team members. One interview revealed that for some people, going through the prototype easily opens up what is being done but others might not be so motivated to go through the prototype and might then independently try to understand the system and the bigger picture. One of the interviewed designers interviewed suggests below a bigger internal walkthrough to share understanding:

Giving just notes like "this is what came out from SVS, here is the prototype, go and click, read and understand". I don't believe in it. There should be some bigger internal walkthrough and the whole sharing the understanding in a different level – Designer 1

Problems were also found if some members of the development team had not been briefed on SVS outcome well enough before the start of the customer kickoff and then not being completely aware of all of the choices made in the SVS. Interviewee mentioned that if team members were briefed well in advance before the customer kickoff, strange debates about for example technical choices could be avoided with the customer. In the following quote, the interviewed designer explains how briefing people in advance avoids confusing situations in customer kickoffs:

The job would be very easy to tackle if the people had been put into that project well in advance and everything is aware of what is being done and why some technical choices have been made so and so. Then there wouldn't be such strange debate as to why these technology choices are made. -Designer 1

According to the interviewees, transferring the information is an important part of designer's job, because often the designer who has been doing the SVS is doing the handover. Interviewees stated that there is not a clear phase with only design and then the implementation phase starts. One designer interviewed pointed out that because designers are not able to produce all of the necessary design, for example UX, for the developers to work on, the outcome is usually a concept, maybe some wireframes or prototype that has been tested with people but never the whole thing cannot be given to the developers and chunked into sprints.

Interviewees revealed that putting insight, business problems and user problems to as concrete level as possible could help in creating shared understanding among the project team and with the customer. Cutting the outcome of the SVS into so-called bite size sections in the definition of the MVP, helps the production team to start implementation more effectively. One of the designers told that in an ideal world, the MVP is defined so well it is easy to start breaking it down into technical elements. Other designer describes below what is needed for the developers so what the outcome could be chunked into sprints more easily:

> *We get the general idea of MVP in the SVS, but in order to start development, the developers need something more. They need some kinds of wireframes, prototypes and UI-design to work through and we do not necessarily have time to do that in SVS. So, the outcome is the definition MVP, but on the conceptual level like user stories, epics and this kind of stuff.* – Designer 2

4.5 Supporting Agile Practices

Supporting Agile Practices is the fourth main theme. This subsection highlights the interview topics that were seen to have an impact on the success of an Agile success project.

4.5.1 Team Environment

This thesis was conducted during a global pandemic Coronavirus (COVID-19) and most of the people around the world were teleworking to avoid infections. Even though teleworking is not new to the software industry, it still could bring some challenges to everyday working. While teleworking, one interviewee mentioned that most of the coding in their project was done in so-called remote mob programming style where the whole team is working on the same issue but remote. The developer below explains the normal way of working for their team while everyone is remote:

> *That means opening the video call, someone coding and others commenting and solving the problem, discussing that code and its further development. That's the normal way of working at the moment.* -Developer 1

Remote working brought some challenges to communicating with team members. The usual daily Scrum was, according to some interviewees, more challenging when people did not see face-to-face. The quote below tells how it is more challenging to notice if everyone is getting heard in the video call:

Face to face is better because you can see how that coworker is feeling in that moment and whether everyone has been listened to. In a remote video conference, it's harder to notice if something hasn't been listened to or something has gone by or you may not hear everything anyway" -Developer 1

Before so-called Corona time, some designers used to work with the clients in a physical place with the Lean Service Creation canvases. When teleworking became the norm, designers and developers started to quickly switch to Miro. Interviewed designer told that to maintain maximum transparency so that everyone always sees what everyone else is doing, they started to do work with the developers and customers in Miro and create shared Miro boards. The quote below explains how collaboration works conveniently between interviewed the designer and the developers in the SVS using Miro:

There we see what everyone is doing. For example, developers are starting to sketch the system's architecture and the same time we are sketching scenarios and prototypes and things like that in the same Miro board. We are doing these workshops at the same time for Service Vision Sprint. -Designer 2

4.5.2 Leadership style

Leadership style is the third subtheme for Agile practices. It was seen crucial that the team was allowed to remain self-directed and the possible team leader acted in a servant leader role. According to the interviews, these types of leaders were also valued the most. Good leaders were seen to inspire people, getting more out of then voluntarily, and setting an example themselves. The interviewed facilitator and the interviewed developer below emphasize the importance of an inspiring team leader for the success of the project:

There are so many pros. People are more efficient, employee engagement is much higher, and they have the market evidence, so my assumption is they make better business decisions. And if they know the business context that maybe only the manager sees, it needs to be a communication and two directional streams of information. – Facilitator 1

Projects work best with a committed and enthusiastic team and a Scrum Master who inspires people and sees Scrum as a tool for the outside world and not specifically for team leadership. -Developer 2

The interviews revealed that the management style in the case company is more focused on the high-level budget role and managing that everything stays in schedule. The leadership style is otherwise so called "when needed" and usually teams have a team lead who runs the project and has the best view in the situation. One of the designers told that the project manager comes usually from the client's organization.

The interviewees told that the teams strive to keep the hierarchy low. One interviewed designer told that usually there is a project owner and someone who is nominated to act as an Agile leader. This person leads the sprint planning and makes sure everything is in place, the others make decisions equally.

One of the interviewed designers shared a good experience working in a client case, where their team was the first team in the customer company to which was tried to bring Agile ways of working with a project manager from the customer side. This project manager knew Agile ways of working and was able to strike a balance between a hierarchical way of working and agility. The interviewed designer and the interviewed developer below describe how the job of the leader is also to protect the team:

> *This* [Agile project manager from the customer side*] project manager acted kind of an "umbrella" for us.* – Designer 3

> *[Leader] Acts itself as a shield for the team and clenches the teeth alone. Gives the team peace of mind and leaves politics out of it.* – Developer 2

Communication

Interviewees told that the client's work environment may be more hierarchical, and it is not sometimes possible to ask everyone everything. Some things go first through a higher-level person who could be extremely busy. The interviewees discuss below how the ways of working and communication style may differ with the customer:

> *At times, the client might have a board making decisions, but we never got to communicate with them because there was always some intermediary.* -Developer 2

> *We wouldn't have to work as much on the client's premises if we could catch people easier.* -Designer 3

One of the interviewees told that problems in daily communication rarely come between the case company team members. Designer mentioned that one thing that prevents information sharing on the case company's side for design is overlapping projects. In the following quote the interviewed designer mentions what challenges overlapping projects bring in terms of communication:

> *When you're doing SVS with reduced hours, at times you just feel like you don't have enough time to go next to some developer and go through things.* -Designer 3

The Lean Service Teamwork canvases also aided in team communication and helped to better understand other team members. One interviewee told that the Weekly Smileys from the Teamwork canvases is an extremely good tool to understand the psychology of your own team. Teamwork canvases can be found on Appendix B. The interviewed developer discusses below why weekly smileys should be brought to every project possible:

> *Weekly smileys are really good, as care is taken to ensure that everyone on the team has their own voice and everyone's well-being is taken care of. Sometimes it's pretty good to stop and ask, "how are you feeling". Especially now on Corona time.* - Developer 1

4.6 Stakeholders

Stakeholders is the fifth main theme. This section goes through interview results that highlight how customer involvement and end user involvement affect the success of the project. One of the core values of Agile manifesto stands for customer collaboration over contract negations.

4.6.1 Customer role

Based on the interviews, the client's participation in the project is always case-specific. Co-designing with the client was seen as a positive thing. It was also seen good that clients are willing to participate more stakeholders from their company to the SVS process. Still it was seen important that there should be some core group with decision-making power and involve others to gather understanding from them. The aim is to get those stakeholders who have some influence on the decisions to get their say and keep them informed of what is being done. Sometimes customers participate less that the team wishes. The interviewed facilitator below explains why involving customer more in the process is not always possible:

Way less than I like them to be [customer involvement]. *It is quite common that customers would be used in the beginning in the interviews but would be rarely integrated into the process. That is something that is sometimes very hard to convince organizations to do because there is hesitance to show any product that isn't finished to the customers.* – Facilitator 1

The interviewed designer suggested that clients should be able to brainstorm with the concept not only on a concrete level but with concrete things and what things should look like. The interviews revealed that some projects have used printed views from similar applications and then given them to customers to examine. Customers are then allowed to circle what looks nice about them and what kind of structures and interactions they prefer. The interviewed designer mentions below how ready set pictures could ease the designer's job:

Of course, it's time consuming to look for some pictures and it's faster to just take the canvases. But if we had some ready set pictures, it could make things so much easier. -Designer 2

4.6.2 End user

Found from the interviews, the role of the client in the workshops is to share thoughts and desires they have for the service. Sometimes people from the customer side involving the SVS still are not the ones who are using the end product and are acting as a messenger to end users. Involving end users in the testing phase was seen as useful for further development. The interviewed developer below describes how workshops and pilot testing with the end users are used to build understanding of the user needs:

I really like a phase where there is a controlled pilot where we invite a certain number of real users to try that product maybe for free or at a reduced price where they can give direct feedback on the usage from which we can then iteratively start developing that pilot all the way to a full launch. – Developer 1

There was a general desire to involve end users in the design phase and to test the pilot. Depending on the project, sometimes customer and end user might mean the same thing. If some in-house service is being built, the end user might be a stakeholder within the company, an employee to whom the tool is made. The aim is to make their voice heard in the SVS. The aim is also to make use of the technical knowledge the company has inside the house and involve those in the process because usually they have the knowledge

of the current systems. The following quote describes who important it is to get a widest possible sample of who is involved in the customer's processes:

> *Views from somewhere e.g. customer service, it, business, they may be very different from each other. We do this with everyone with the canvases.* – Designer 1

5 Discussion

This thesis investigated how the Lean Service Creation method could be improved in order to make the handover more effective between Service Vision Sprint and implementation phase. The aim of the study was to find out what are the challenges and success factors in the handover between Service Vision Sprint and the implementation phase and how the handover could be improved. The following sections present the research results linked to previous studies, the thesis process review and the future development ideas.

5.1 Results and Literature review

This thesis shows that handover between SVS and implementation phase can vary a lot depending on the project. Transition phase or handover was said to work well generally, but suggestions for improvement to improve the process were also given. Suggestions to improve SVS were also given, but they are also much related to handover.

This thesis was able to determine that the biggest success factors in the transition from SVS to implementation phase and Agile Software Development are strongly related to shared understanding, internal kickoffs and different roles of the development team and stakeholders. The material obtained from the interviews also supported the success factors found in the literature review.

Interviewees often raised a shared understanding or overall understanding as a motivator for starting a new project. This thesis' results show that a shared understanding increases when all members of the project team understand the basic needs behind the service, as well as an understanding of business and the industry. Internal planning sessions and internal kickoffs were seen as one of the most important actions to increase overall understanding of the project and to motivate team members who are onboarding the project. Based on this thesis' results, internal walkthroughs were seen important for sharing understanding because the prototype that is usually the outcome of the SVS, does not always fully clarify what is being done to the developers onboarding the project. However, a good prototype was seen as a good tool for getting in touch with what the end product will look like and what is the user's need. This thesis' results revealed that an internal review of SVS outcome is particularly important before a client kickoff, so that everyone onboarding the team is aware of what is being made and why some technological choices are made. This thesis' results are very much in line with the study of Kelle et al. (2015). Communication style and value congruence were found to be one of the critical success factors in Agile Software Development. Informational communication helps to build trust, creates more shared values and creates strong interpersonal relationships. If goals and targets differ between the members, it can decrease satisfaction and negatively affect software's performance. (Kelle et al., 2015) These research findings are also in line with

the fifth principle from LSC Manifesto (Nevanlinna et al., 2018). The fifth Principle *See the Forest, See the Trees, Spot the Squirrel,* underlines the importance of seeing the bigger picture for overall understanding.

This thesis' results show that a thorough technical background study is important during the SVS in order to be able to build sensible architectural solutions. A similar conclusion was reached by Poppendieck and Poppendieck (2003). Even though sticking to a plan makes a software weak, planning is needed to create high-level architecture design and it still makes a good learning experience (Poppendieck & Poppendieck, 2003). Study of Jain et al. (2018) is also in line with this thesis's results, since it states that architectural design is part of requirement analysis, which is part of Agile Software Development process.

Suggestions for improvement were also given on how the outcome of the SVS should be documented. This thesis' results revealed that concepts or presentations of the SVS outcome were sometimes left fairly abstract, which could leave them unclear to UX-designers and developers. This thesis' findings suggest that the documentation could be improved if insight, business problems and user problems were brought to the most concrete level possible in the definition of MVP. This contradicts previous studies in that they do not emphasize the importance of documentation other than in terms of code documentation. These research findings are also in line with the third Principle from LSC Manifesto (Nevanlinna et al., 2018). The third principle, *No matter what you do, be transparent about it,* which underlines the importance of making decisions and rationale understandable and abstract things tangible.

The results of this thesis indicate that a strong hypothesis is important in the handover. A strong hypothesis helps the team to know what the purpose of the service or product is and what they are trying to experiment and learn. It was seen that with a strong hypothesis, the team can build and prioritize their work more accordingly. This thesis' results show that without a strong hypothesis the MVP might come out weak, which could cause the scope to grow and delay the delivery. This thesis's findings are in line with the study of Chow & Cao (2008), that found the right delivery strategy to be critical success factor for Agile Project success.

The results of this thesis indicate that involving developers and UX designers as much as possible in the SVS increases the meaningfulness of their work, as well as helps them to better understand the goals and the vision. This thesis's results show that meeting end users in the SVS workshops is an excellent way to get in-depth background information

about the user needs. This thesis' findings are in accordance with the findings reported by Denning (2013). One of the core principal values of Design Thinking is a team with diversified experience and tangibility (Denning, 2013).

This thesis' results show that it is important to transfer the sense of ownership to the team in the handover, which is in line with the Lean Service Creation Handbook (Futurice LSC Handbook, 2019). This thesis' findings indicate that the sense of ownership strengthens if the team is actively steering the direction of the project, being involved in creative building and shaping the idea, and the team is let to carry the vision of the project. This thesis' results tie well with the previous study of Poppendieck & Poppendieck (2006). Projects that undergo constant changes and respond to changes in the market are more likely to be successful than projects that strictly follow the early requirements (Poppendieck & Poppendieck, 2006). This thesis' findings are also in line with the first principle from LSC Manifesto (Nevanlinna et al., 2018). The first Principle *All for the Team, Team for All,* underlines the importance of care and trust in teams. In order for creativity to flourish in multidisciplinary teams, it requires systematic teamwork to find team's own best practices and routines to work (Nevanlinna et al., 2018). This thesis' results are also in line with the fourth Principle of the LSC Manifesto, *Never Stop Iterating, Never Stop Learning* (Nevanlinna et al., 2018). It is always good to leave room for iteration and a detailed project plan forces the team to focus on micro level targets that are far from reality. (Nevanlinna et al., 2018) Similar findings compared to this thesis are shown in the study of Chow & Cao (2008). Team environment is Agile if teams remain small, coherent and self-organizing. Projects should not have multiple independent teams and teams should have the collocation of the whole team (Chow & Cao, 2008).

This thesis found clear support for Agile methods in LSC projects. This thesis' results show that teleworking brought some challenges to communicating with team members. Teleworking made communication more difficult in meetings, as all participants may not be heard in video calls and people's general mood is not noticed. These are in line with the study of Kelle et al., (2015) that states the lack of effective commutation and misunderstandings to be one of the main reasons for failures. A similar finding compared to this thesis was reached by Kelle et al. (2015). Communication style is one of the critical success factors in Agile Project development. Informational communication helps to build trust and helps to create more shared values and interpersonal relationships. (Kelle et al., 2015) Face-to-face communication to spread information to and between team members is also one of the principles behind Agile Manifesto (Agile Alliance, 2001). This thesis results show that teleworking was adapted well in the case company and new ways to work effectively remotely was found both between the team and the customers.

The results of this thesis revealed that the management style on the case company side is more focused on the high-level budget role and managing that everything stays in schedule. Each team has either an Agile leader or a team lead, but otherwise everyone has an equal say and hierarchy is kept low. This thesis' results indicate that controlling management style could hurt the sense of ownership and a servant leader style has a big impact on the success of a software project. This is in line with earlier studies of Schwaber and Sutherland (2017) and Poppendieck and Poppendieck (2006). Scrum Teams should be self-organizing, and the team decides the ways to deliver the functionalities. Teams who have engaged people who are trusted and are able to make decisions without permissions can respond to customers' wishes faster and improve their processes. (Schwaber & Sutherland, 2017; Poppendieck & Poppendieck, 2003) This thesis' findings are also in line with the first principle from LSC Manifesto, *All for the Team, Team for All* (Nevanlinna et al., 2018). Hierarchies and airtight social groups are a great barrier for innovations and creativity (Nevanlinna et al., 2018). Similar findings compared to this thesis was also found from the study of Kelle et al. (2015). Transformational leadership is one of the critical success factors for Agile projects. Leaders who are this type, motivate, inspire, express visions and engage their followers with emotional involvement (Kelle t al., 2015).

The results of this thesis provide evidence that co-designing with the customer is beneficial for understanding the stakeholders and end users better. This thesis' results show that involving end-users as much as possible in design and testing was seen as beneficial for future development. Direct feedback from end users help the teams to start iteratively developing the pilot to a full launch. This thesis' findings support the study of Denning (2013). One of the principal values of Design Thinking is Tangibility. Tangibility means learning from feedback and reactions after customers have tested prototypes. (Denning, 2013) This thesis' results are also in line with the study of Abrahamsson et al. (2002), and Chow & Cao (2008). Development model is Agile if the customers and the developers are cooperating continuously. Strong customer involvement is also a somewhat critical success factor for Agile projects. (Abrahamsson et al., 2013; Chow & Cao, 2008)

5.2 Key conclusions

Service Vision Sprint

- Resolving technical matters as soon as possible, since the most important things are found on the technical side.
- Possible prototype stage determines what can be iterated before building anything extremely tangible and big into MVP stage.
- Building a good prototype motivates the team members joining the project.

- Meeting end users and stakeholders in SVS workshops helps UX-designers and developers to understand the end user needs and makes the vision clearer.

Handover

- The level of documentation of the SVS outcome must be concrete enough, so that everyone onboarding the project understands the insight, business problems and user problems. If the SVS documentation is too abstract, it might be less inspiring for people onboarding the project.
- If the customer has a clear picture of what they want, concrete ideas should be gathered during SVS into a canvas that bites deeper into the concept.
- Teams should have someone with an Agile background who can write strong hypothesis. A strong hypothesis predicts the MVP coming out weak and helps the team stay inside the scope.
- If the transition period between the SVS and implementation phase is too long, the information may become outdated and it is harder for people being involved in the SVS to remember all the details.
- If internal kickoffs are structured well, they offer very good specifications for development work. Only one internal kickoff was not seen to be enough for sharing understanding.
- Team should carry the ownership of the product by actively steering the direction, rebuilding assumptions and be focused on validating.

Shared Understanding

- Understanding the big picture is the biggest motivator for people when starting a new project. When people understand the entire business, industry and user problems, and are on the same page, the work becomes more meaningful.
- Outcome of the SVS should be briefed to everyone in the project always as good as possible before customer kickoffs to avoid strange debates for example about technological choices.
- Bigger internal walkthroughs of the SVS outcomes helps to increase the overall understanding.
- Cutting the outcome of SVS into bite size sections in the definition of the MVP, helps the production team to start implementation more effectively.

Supporting Agile Practices

- Face-to-face communication eases to see how team members are feeling and if everyone has been listened to.

- Servant leader type of leadership style that allows the team to stay self-directed makes people more efficient.
- Weekly smileys -canvas is a good tool for understanding the psychology of the team.

Stakeholders

- Customers should be able to brainstorm with the concept not only on a concrete level but with concrete things and what things should look like.
- Involving end users in a controlled pilot phase increases the understanding of the user needs. Direct feedback of the usage from end users helps to iteratively develop the pilot to full launch.

5.3 Thesis process review

The thesis work started at the beginning of June 2020 and it was finished in the middle of October. Thesis camp provided by the case company supported the thesis work for the first two months by providing guidance and time for the writing process.

The interview process was the most interesting part of this thesis, since I have not been involved with the Lean Service Creation process while working in the case company. Lean Service Creation as a whole has been of interest to me, which was one of the reasons for doing this thesis. Interviews also gave insight to the work of Service Designers and UX designers which was highly interesting since I work as a software developer.

A limitation of this thesis is that the numbers of interviewees were small. Finding interviewees within the case company also brought challenges to the interview process, as many did not believe that they were the right target interviewees for the research question. Tight schedule and rapid analysis of the interview material can also be seen as a limitation of the study. In order to obtain more accurate and extensive research results, it would be a good idea to conduct this thesis with a larger number of interviewees.

Since most of the interviewees worked in Finland, it would be interesting to involve more people working in foreign case company offices and examine how working with the SVS process differs between countries.

Translating from Finnish to English also brought challenges to the transcription of the interviews, as the message and feeling of the quotations were to be kept similar to the original language. During an interview in English, it also became clear that the word

handover is clearer when compared to the word *transition*, so the body of the interview was modified as needed.

The role of canvases was left smaller in the interviews than expected. Also, in some interviews, there was not that much time left for a thorough review of the canvases.

It was surprising to how much the actions in the SVS affected the handover and the starting of the implementation phase, and not so much the things that happen in the handover. Surprisingly, the interviews focused a lot on the SVS itself, which, according to the material obtained from the interviews, had a great impact on the success of the handover.

I would like to thank the case company for the opportunity to conduct this thesis. The overall research process has been highly interesting and hopefully the case company will benefit from the findings of this research.

5.4 Future development ideas

From the material obtained from the interviews, it was possible to extract a couple of ideas which could help improve either the SVS or the handover.

Because some of the interviewees felt that the concepts of SVS remained often fairly abstract, some new SVS canvas that could bite even deeper into the concept could be helpful if the client already has a clear vision of what they want. One designer brought up the idea of a canvas that could be used to gather all the concrete ideas of the concept for the UX-designers and developers to start their work more efficiently.
Otherwise, I would stress that future projects would pay special attention to shared understanding between the team members, so that the level of motivation is the highest possible when onboarding a new project.

6 Conclusion

It is easy to see why Lean and Agile methods are increasingly being applied in the software business. Lean and Agile methods enable the development of high-quality software quickly and according to user wishes. The use of Lean and Agile methods makes it possible to respond to changes in the market faster which is crucial for a successful software project in a rapidly changing world. Lean Service Creation is a service design and development process that has developed around Lean Startup, Agile Development and Design thinking. The aim of the study was to find out how the handover between Service Vision Sprint and implementation phase could be improved in the Lean Service Creation process. The study also wanted to find out how the success factors of an Agile software project found in previous research were applied in LSC projects. To get suggestions for improvement, this thesis interviewed six people, who have been involved in Lean Service Creation projects, and at the time of conduction of the interviews, were all employed by the case company.

The results of this thesis are very much in line with previous studies collected in the literature review, but some differences were found. The biggest similarities are related to the importance of communication style and self-directed teams. The interviews highlighted the importance of documentation in every aspect of the project, but in the literature review, documentation was only highlighted in the documentation of the code. A key result of this thesis was the importance of common understanding among the team members of the SVS outcome and sharing understanding in the handover. Other key findings were the quality of documentation of the SVS outcome, the importance of technical background work, sense of ownership, collaboration of designers and developers in the SVS, inspiring leadership style and actively involving the end user.

Handover between the SVS and the implementation phase could be further explored, although more extensive research on the integration of design and development has already been done in the past. However, only one previous study that focuses on LSC has been done at the time of this writing this thesis, so research topics around it could be formed for example from a service design perspective.

7 References

Abrahamsson, P., Salo, O. and Ronkainen, J. 2002. Agile software development methods : review and analysis. VTT Publications 478, Espoo.

Agile Manifesto. 2001. Manifesto for agile software development. Retrieved from: http://agilemanifesto.org/ (Used: 4.4.2020).

Agile Alliance. 2001. 12 Principles Behind the Agile Manifesto. Retrieved from: http://agilemanifesto.org/principles.html/ (Used: 4.4.2020).

Agile Report. 2020. State of Agile: 14th annual state of agile report. Digital.ai. Retrieved from: *https://stateofagile.com/home* (Used: 30.7.2020)

Atlassian. 2020. Atlassian: What is a kanban board? Retrieved from: *https://www.atlassian.com/agile/kanban/boards* (Used: 12.4.2020).

Bacea, I. M., Ciupe, A. and Meza, N. (2017). Interactive kanban — blending digital and physical resources for collaborative project based learning. 2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT), pp. 210-211. Retrieved from: https:// doi:10.1109/ICALT.2017.68

Brem, A. and Fredriksen, D.L. 2016. How do entrepreneurs think they create value? A scientific reflection of Eric Ries' Lean Startup approach. The International Entrepreneurship and Management Journal, 13(1), pp. 169-189.

Chow, T. and Cao, D. B. 2008. A survey study of critical success factors in agile software projects. Journal of Systems and Software, 81(6), pp. 961-971

CBT Nuggets. 2020. CBT Nuggets: How Do Kanban, Scrum and Lean Relate? Retrieved from: *https://www.cbtnuggets.com/blog/career/management/how-do-kanban-scrum-and-lean-relate*. (Used: 15.7.202)

Clemente, V. and Tschimmel, K. (2016). A learning toolkit to promote creative and critical thinking in products design and development through design thinking. 2nd International Conference of the Portugese Society for Engineering Education (CISPEE), pp. 1-6. Retrieved from: https:// doi:10.1109/CISPEE.2016.7777732

Kassab, M., DeFranco, J. and Graciano Neto, V. (2018). An empirical investigation on the satisfaction levels with the requirements engineering practices: Agile vs. waterfall. IEEE International Professional Communication Conference (ProComm), pp. 118-124. Retrieved from: https:// doi:10.1109/ProComm.2018.00033

Denning, P. J. 2013. Design thinking. Communications of the ACM, 56(12), pp. 29–31.

Futurice. 2020. Futurice Oy: Services. Retrieved from: https://futurice.com/services (Used 5.4.2020)

Futurice LSC. 2020. Futurice Oy: Lean Service Creation. Retrieved from: *https://futurice.com/lean-service-creation/* (Used: 5.4.2020)

Futurice The LSC Handbook. 2019. Futurice Oy: The Lean Service Creation Handbook. Retrieved from: *https://hello.futurice.com/lsc-handbook* (Used: 5.4.2020)

Jain, P., Sharma, A. and Ahuja, L. (2018). The impact of agile software development process on the quality of software product. 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 812-815. Retrieved from: https:// doi:10.1109/ICRITO.2018.8748529

Janes, A. (2015). A Guide to Lean Software Development in Action. 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Graz, 2015, pp. 1-2. Retrieved from: https:// doi:10.1109/ICSTW.2015.7107412

Kanban University. 2020. The Kanban Method. Kanban University. Retrieved from: *https://edu.kanban.university/kanban-method* (Used: 12.4.2020)

Kelle, E. V., Visser, J., Plaat, A. and Wijst, P. v. d. 2015. An empirical study into social success factors for agile software development. 2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering, pp. 77-8

Nakazawa, S., Komatsu, K., Tanaka, T. and Matsumoto, K. 2017. Development and evolution of large-screen digital Kanban with smartphone operation. 6th IIAI International Congress on Advanced Applied Informatics (IIAI. AAI), pp. 295-300. Retrieved from: https:// doi:10.1109/IIAI-AAI.2017.151

Kurnia, R, Ferdiana, R and Wibirama, S. 2018. Software metrics classification for agile scrum process: A literature review. Paper presented at the International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), pp. 174-179. Retrieved from: https:// doi:10.1109/ISRITI.2018.8864244

Lulu, M. 1986. Just-in-time (JIT) production and process unreliability. In Proceedings of the 19th Annual Symposium on Simulation, Tampa, Florida, USA, pp. 237–249.

Rodriguez, P., Markkula, J., Oivo, M. and Turula, K. 2012. Survey on agile and lean usage in Finnish software industry. International Symposium on Empirical Software Engineering and Measurement (ESEM'12), Lund, Sweden, s. 139-148. Retrieved from: https://doi:10.1145/2372251.2372275

Nakata, C. and Hwang, J. 2020. Design thinking for innovation: Composition, consequence, and contingency. Journal of Business Research, 118, pp. 117-128. Retrieved from: https://doi.org/10.1016/j.jbusres.2020.06.038

Nakazawa, S. and Tanaka, T. (2015). Prototype of kanban tool and preliminary evaluation of visualizing method for task assignment. International Conference on Computer Application Technologies, pp. 48-49. Retrieved from: https://doi:10.1109/CCATS.2015.21

Nevanlinna, H., Sarvas, R. and Toiminen, M. 2018. Open Source Tools for Change Agents – the what, the how and the why. Printed in Estonia, 2018. Futurice Oy.

Poppendieck, M. and Poppendieck, T. 2003 Lean Software Development: An Agile Toolkit. Boston: Addison Wesley Professional

Poppendieck, M., and Poppendieck, T. (2006). Implementing Lean Software Development: From Concept to Cash. Addison Wesley Professional.

Lean Enterprise Institute. 2020. Lean Enterprise Institute: Principles of Lean. Retrieved from: *https://www.lean.org/WhatsLean/Principles.cfm* (Used: 5.4.2020.)

Razzak, M. A. (2016). An empirical study on lean and agile methods in global software development. IEEE 11th International Conference on Global Software Engineering Workshops (ICGSEW), pp. 61 – 64 Retrieved from: https://doi:10.1109/ICGSEW.2016.22

Schwaber, K. and Sutherland, J. 2017. Scrum Guide. Retrieved from *https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf* (Used: 13.4.2020).

Scrum Alliance. 2020. Overview: What is scrum? Scrum Alliance. Retrieved from: *https://www.scrumalliance.org/about-scrum/overview* (Used: 13.4.2020).

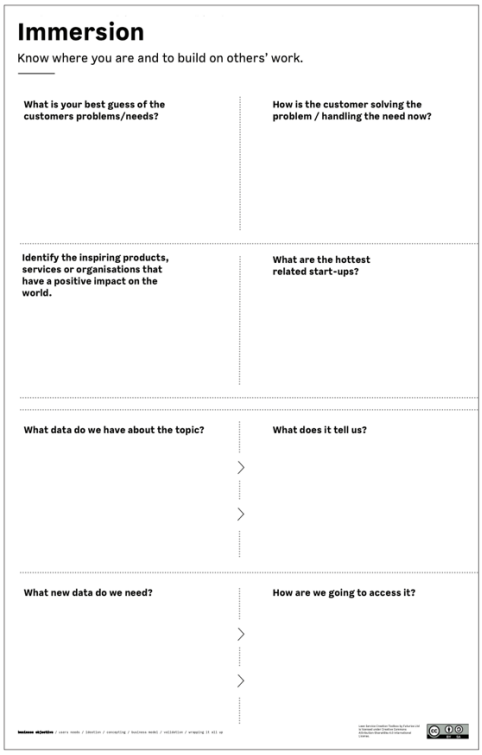
The Lean Startup. 2020. The Lean Startup: The Movement That is Transforming How new Products Are Built and Launched. Retrieved from: *http://theleanstartup.com/* (Used: 1.7.2020).

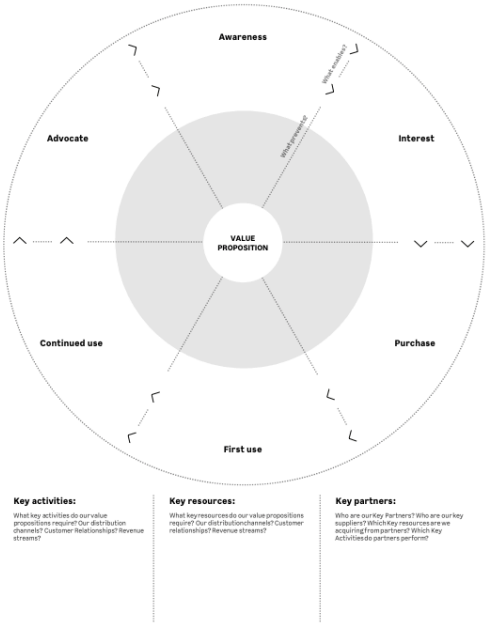
Veretennikova, N. and Vaskiv, R. (2018). Application of the lean startup methodology in project management at launching new innovative products. IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), pp. 169-172. Retrieved from: *https://doi:10.1109/STC-CSIT.2018.852673*

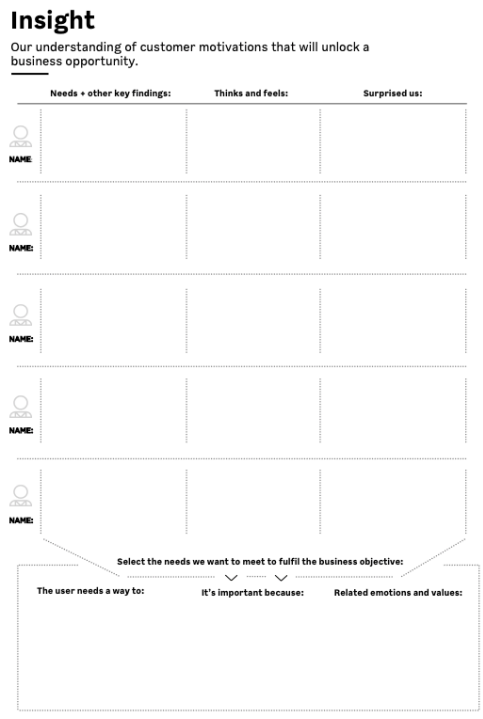
Womack, J and Jones, D. 1996. Lean Thinking: Banish Waste and Create Wealth in Your Corporation. New York, NY: Simon and Schuster

Wykowski, T and Wykowska, J. (2018). Agile Alliance: Lessons Learned: Using Scrum in non-technical teams. XP 2018 Conference. Retrieved from: *https://www.agilealliance.org/resources/experience-reports/lessons-learned-using-scrum-in-non-technical-teams/* (Used: 13.4.2020.)

Appendix A: Lean Service Creation Canvases







Script Creator

Preparing for the insight interviews.

Don't ask what the user wants, try to find out what she needs on the deeper level.

What do we need to learn about the interviewee?

How to ask about it in the interview?

How to ask about it in the interview?

How to ask about it in the interview?

How to ask about it in the interview?

Ask; "Tell me about the last time you..." and draw a timeline with the customer.

A good way of conducting interviews is to draw the timeline/journey with her on a piece of A3 paper, ask a lot of 'why' questions and then seek to understand the emotions and values related to the events. Fill this template in first, then write a script out of it and try it out.

1. Which phases of the customer's journey are related to your business?

2. What do we need to learn about the actions and motives related to these phases?

3. What do we need to learn about the emotions and values related to these phases?

Intro

? How to introduce yourself and the interview?
? How will the answers be handled?
? How to ask the basic info?
? How to categorize the person belongs to the select customer group?
? What kind of warm-up question would help us to understand the everyday life of the interviewee?

Thank you

? How will you reward the interviewee? How will you end the conversation? How will you follow the time? Any ending questions;
? Who is the nobody else we should talk in your organisation?
? We are working on solving these problems, is it OK to interview you again?

How will you document the interview?

Agree on the practicalities with your team. Remember to always write down the things that surprised you!

ASK WHY.

Customer Grouping

Your best guess of who you aim to serve.

Identified customer need: | **Description:** | **Group name:**

Identified customer need: | **Description:** | **Group name:**

LOVE THE PROBLEM
NOT THE SOLUTION

Main Group: ☐

Identified customer need: | **Description** | **Group name:**

Main Group: ☐

Common in all user segments:

Ideation Sandbox

To kickstart the first round of ideation

User need/problem

Write it down on a high enough level so that your idea is not the only solution available.

Emotions and values

Related emotions and values from the interviews.

Purpose of the service for the user

What job does the service do for the user? Take the needs, emotions and values and write down the main purpose of the service for the user.

Business Objective

What is the objective of the service for your company?

Company Mission

What is the mission of your organisation?

Team Mission

What is the ambition of the team? What do you want to achieve?

Purpose of the service for the society?

How should the service help the society and environment?

Ideas that fill the user need, business objective and/or society purpose

After filling the sides of the sandbox start to ideate.

Idea Accelerator

To make the most out of the selected seed idea

Seed idea

Select the best idea from the ideation Sandbox canvas as the seed idea.

Three best ideas to make the concept draft (seed idea?) better fulfil the user need

Behavioural gap

What is the behavioural change the user needs to adapt to use the service?

Three best ideas to help the user over the gap. Try to change the core of the idea. Don't just add a manual...

Business objective

Regarding the business objective where does the concept draft fall shortest?

Three best ideas to make the concept match the business objective better.

Social purpose

What is the social and environmental impact of this idea? How is it compared to your goal?

How could the concept have a bigger and better social impact or be more sustainable?

Accelerated idea

Write down the holistic concept that has been created based on these new ideas.

Are you ready to proceed? Or should we redo this canvas by using the accelerated concept as a seed idea?

NO! ⟵ YES!

Rational Concept Sheet

Draft a concept out of your idea

Concept name?

How does it work?

Value to the end-user?

LOVE THE PROBLEM
NOT THE SOLUTION

What differentiates it from other solutions to the same problem?

Value to our business?

Social and environmental impact?

Rational value proposition:

END USER:

NEED:

SOLUTION:

Concept Sheet

Adding the emotional and value level to the concept

Negative emotions?

What are the negative emotions related to the needs of the customer?

How is the concept helping with the negative emotions?

Positive emotions

What are the positive emotions related to the needs of the customer?

How is the concept making the most out of the positive emotions?

Value gap

What is the gap between customers values and current actions?

How is this concept helping to fill the value gap?

Emotional value proposition:

END USER:

EMOTIONAL NEED:

SOLUTION:

Combine both value propositions here:

What is the mental reward for the customer?

Fake advertisement – The first prototype:

Create the first prototypes of your service by creating a fake ads. You can use them to test your value proposition. Use blank paper as template. The following ad template is a good starting point.

PICTURE:

CAPTION:

HEADLINE:

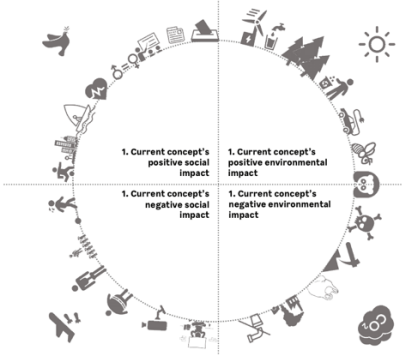
COPY:

Impact Optimiser

Enhance the good impact and minimise the bad. Ideation continues!

2. Tweak the concept to maximise the positive societal impact

2.Tweak the concept to maximise the positive environmental impact



1. Current concept's positive social impact

1. Current concept's positive environmental impact

1. Current concept's negative social impact

1. Current concept's negative environmental impact

2. Ideas to reduce the negative social impact

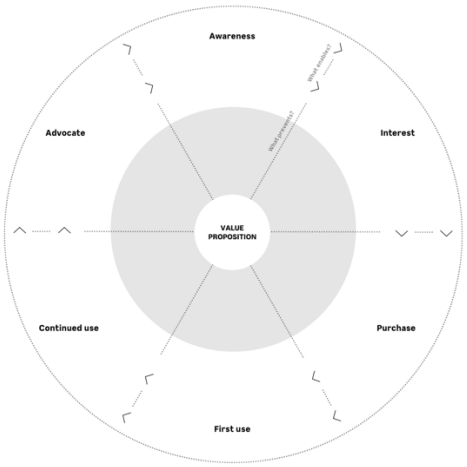
2. Ideas to reduce the negative environmental impact

Is our impact net positive to the world?

NO! — Hard to say — YES!

Customer Engagement

How do we get people to become our advocates? Ideation continues!



Key activities:

What key activities do our value propositions require? Our distribution channels? Customer relationships? Revenue streams?

Key resources:

What key resources do our value propositions require? Our distribution channels? Customer relationships? Revenue streams?

Key partners:

Who are our Key partners? Who are our key suppliers? Which Key resources are we acquiring from partners? Which Key activities do partners perform?

Business Model & Market Size

Thinking through the high level business case.

Who pays whom? How much?

What is the revenue model?

How are customers paying today? How much?

Where and how do you anchor price perception?

Total addressable market:

1st year sales:

Potential target market:

Plan for 1st month sales:

Cost structure?

Key elements for EBIT?

Feasibility Study

To make sure we can make this happen

Biggest implementation issues?

Potential issue:

What is the most risky tech dependency? What in our organisation is preventing us from success? What is risky in building it?

How to validate:

How can we test this and eliminate the risk?

Findings & solution:

What did we learn and change?

Back to the future:

What will this concept look like in the future?

How to take advantage of it? How to keep an eye on it?

Reduce project risks by simplifying the service:

How could we simplify the service to make it easier to build:

How would this change affect the value proposition?

Evaluating the Concept

Time for the acid test.

Should we proceed?

NO! — WE HAVE TO ITERATE — YES!

Validation

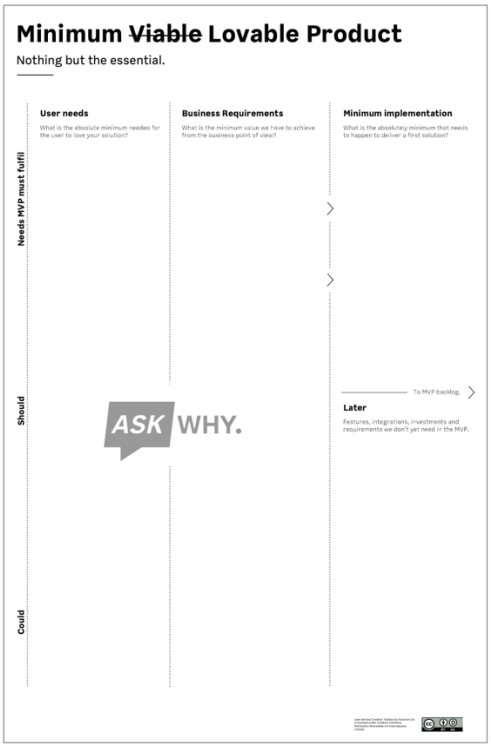
Don't assume, validate. Do whatever it takes to question your concept.

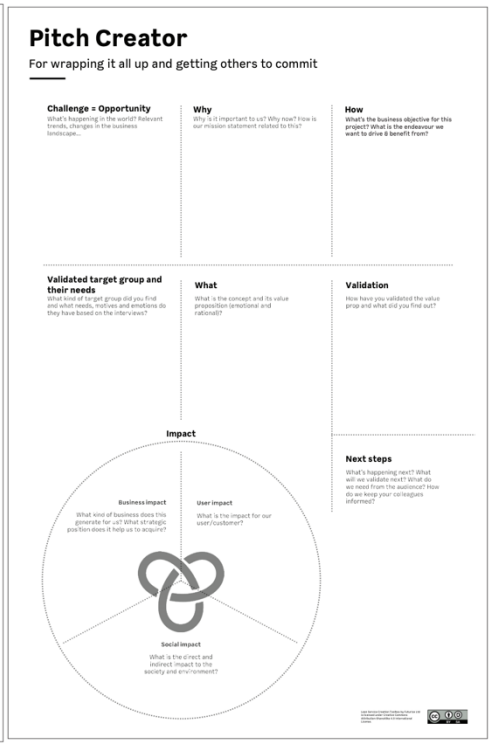
How might we validate our:

VALUE PROPOSITION	CUSTOMER GROUP	BUSINESS MODEL
RESULTS	**RESULTS**	**RESULTS**
CONCLUSION	**CONCLUSION** What size?	**CONCLUSION** How much € ?

Should we proceed?

NO! — WE HAVE TO ITERATE — YES!


Minimum ~~Viable~~ Lovable Product
Nothing but the essential.
User needs
What is the absolute minimum needed for the user to love your solution?
Business Requirements
What is the minimum value we have to achieve from the business point of view?
Minimum implementation
What is the absolutely minimum that needs to happen to deliver a first solution?
Needs MVP must fulfil
Should
Could
ASK WHY.
To MVP backlog.
Later
Features, integrations, investments and requirements we don't yet need in the MVP.


Pitch Creator
For wrapping it all up and getting others to commit
Challenge = Opportunity
What's happening in the world? Relevant trends, changes in the business landscape...
Why
Why is it important to us? Why now? How is our mission statement related to this?
How
What's the business objective for this project? What is the endeavour we want to drive & benefit from?
Validated target group and their needs
What kind of target group did you find and what needs, motives and emotions do they have based on the interviews?
What
What is the concept and its value proposition (emotional and rational)?
Validation
How have you validated the value prop and what did you find out?
Impact
Business impact
What kind of business does this generate for us? What strategic position does it help us to acquire?
User impact
What is the impact for our user/customer?
Social impact
What is the direct and indirect impact to the society and environment?
Next steps
What's happening next? What will we validate next? What do we need from the audience? How do we keep your colleagues informed?

Appendix B: Lean Service Creation Teamwork Canvases